Spatiotemporal Coordination in Wireless Sensor Networks

By

Branislav Kusý

Dissertation

Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

December, 2007

Nashville, Tennessee

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

It is a hard task to enumerate all the people that have motivated and supported me during my graduate studies at Vanderbilt university. Foremost, my gratitude goes to the faculty members, researchers and fellow graduate students working at the Institute for Software Integrated Systems. ISIS is a place where research is driven by tremendous innovation and creativity, with a strong focus on practical applications.

First, with deep gratitude, I would like to thank my thesis advisor Prof. Janos Sztipanovits. Thanks Janos for your enthusiasm and support that you have provided me throughout my graduate studies.

My co-advisor, Prof. Akos Ledeczi, has been a true friend, mentor, research advisor, and formidable soccer oponent. Sensor networks research was only starting when I came and under Akos's leadership it has grown tremendously and earned a wide recognition. In our weekly meetings, Akos provided key technical insights, detailed planning, and set strict deadlines. This resulted in successful field demonstrations of our research as well as the rich body of published literature.

My special thanks go to Prof. Miklos Maroti, whom I worked with intensely from my first year at Vanderbilt. Never before have I met a person with such an incredible energy, dynamics, and unsatisfiable desire to take on the hardest research problems over and over again. Most of the results presented in this thesis, came from our joint work with Miklos. I am especially thankful to Miklos for conceiving the idea of the radio interferometric ranging which I had the pleasure to continue exploring, improving and demonstrating its potential for practical applications. Thanks Miklos for all the support, enthusiasm, and inspiration.

I wish to express my gratitude to my colleges in the sensor networks group in ISIS. The presented extensive experimental evaluation would not have been possible without your help, guys, and I truly enjoyed that I forced you to speak English during our meetings as the only non-Hungarian. Peter Volgyesi designed the signal processing algorithm that analyzes the interferometric radio signal on the motes and contributed to our understanding of multipath radio signal propagation. Our tame genetic-algorithms expert, Gyorgy Balogh, contributed with initial localization and tracking algorithms to evaluate the radio interferometric technique, and contributed to the interferometric range disambiguation. Janos Sallai contributed to our understanding of reactive time synchronization algorithms and helped with the design and implementation of the mTrack system during our summer internship at Oak Ridge National Laboratory. Finally, Andras Nadas, Sebastyen Dora, Gyula Simon, Gabor Pap, and Karoly Molnar contributed to various parts of this thesis and implemented a number of TinyOS middleware services which simplified the data evaluation. Xenofon Koutsoukos directed me to explore Kalman filtering that made the tracking of mobile nodes using RF Doppler shifts possible. I would also like to thank Yuan Xue for providing me with valuable feedback as a member of my dissertation committee.

My gratitude also goes to people from other research institutions. Especially delightful was my collaboration with Prabal Dutta from UC Berkeley who contributed tremendously to our formalization of time synchronization canonical services and standardization of their interfaces. Lambert Meertens from Kestrel Institute contributed to the formal analysis of the errors of interferometric ranging. Vladimir Protopopescu, Frank DeNap, and Shankar Mallikarjun from the Oak Ridge National Laboratory have provided valuable comments on the mTrack system which we have designed and implemented at the lab. Finally, I am indebted to anonymous reviewers and shepherds of our published work for their constructive comments.

Support for this work has been provided in the part by DARPA/IXO NEST program (F33615-01-C-1903), TRUST (The Team for Research in Ubiquitous Secure Technology, NSF award number CCF-0424422), ARO MURI grant SA5212-11087, and a Vanderbilt Discovery Grant.

Abstract of the Dissertation

Large-scale networks of low-power, wireless devices integrated with sensors and actuators have emerged as a new platform that promises to seamlessly integrate computational devices with our environment. Information about various phenomena occurring both in nature and human created environments can be gathered at a fine scale, unobtrusively, remotely, and in a cost-effective manner. Such information can be used to control and manage production facilities, predict and asses natural disasters, or obtain better understanding of animal species or natural processes.

The notion of time and space is fundamental in the context of these wireless sensor networks (WSNs): data is typically gathered at different physical locations and at a different time, thus the interpretation of the data is contingent upon the existence of inter-node synchronization of time and location coordinates. In this thesis, we address both temporal and spatial coordination of WSNs.

We argue that structuring time synchronization protocols into layers and standardizing interfaces of these layers improve adaptability, reusability, and portability of the protocols and help to decrease the complexity and increase the efficiency of their implementation. In our approach, time synchronization protocols are structured in three layers which communicate through well defined application programming interfaces (APIs). Further, we provide implementation, performance analysis, and quantitative comparison of a number of time synchronization services which will help developers to identify the time synchronization needs of their WSN applications.

Despite the considerable research effort invested in the area of node localization, a robust sensor localization is still an open problem today when applied to real world problems. Existing techniques have limited range and accuracy, require extensive calibration, or need extra hardware that adds to the cost and size of the platform. We propose a novel radio interferometric ranging technique that utilizes two transmitters emitting radio signals at almost the same frequencies. The relative distances between the nodes are estimated by measuring the relative phase offset of the generated interference signal at two receivers. We implement interferometric ranging on a low-cost low-power off-the-shelf hardware and demonstrate that both high accuracy and large range can be achieved simultaneously. Furthermore, we present the results of localization and real-time tracking experiments deployed in rural and urban environments, demonstrating that our techniques allow for economical deployments and outperform existing localization services.

Chapter I

Introduction

1.1 Wireless Sensor Networks

Technological advances have made it possible to decrease the size, cost, and energy requirements of computing devices enabling their mass production and unobtrusive deployment. Further research has enabled energy efficient sensors, actuators, and radio chips providing these computing devices with ways to obtain information, coordinate, and react to external stimuli. The potential of tightly connecting the physical world with the virtual world of computers has been recognized and *wireless sensor networks (WSNs)* were defined as large-scale, dynamically changing wireless networks of integrated, low-power, resource-constrained, tiny devices that can be deployed in inhospitable and remote environments.

WSNs promise improved efficiency, monitoring and management of the technology that affects our everyday lives, as demonstrated in the plethora of research proposals and visionary papers [2, 24, 15, 32, 23]. For example, one can envision an improved control of infectious diseases in animals which, as recently demonstrated by Avian influenza, can rapidly spread around the globe. Nowadays, cattle are being raised in such numbers that monitoring the health of individual animals has become impossible and as a consequence, large numbers of them are typically sacrificed at any suspicion of infection. WSNs will make it possible to manufacture coin-sized sensors that can be implanted in the body of these animals [22]. The sensors will monitor physical (temperature, ECG, blood pressure), chemical (pH, toxins), and biological (glucose, protein) properties and could be powered by the motion or thermal energy of animals. Continuous monitoring of toxin levels can detect problems with food or water given to the animals. Temperature monitoring can detect disease symptoms at individual animals in timely manner and would allow to isolate them to stop the further spread of infection. Tracking contacts between animals would enable to identify the group of animals to be isolated/sacrified. Finally, keeping the history of these measurements would allow to track back the animals whose meat caused the health problems, even helping to identify symptoms of unknown diseases.

Although the sophistication required for such infectious disease control system may only become available in 10 years, many WSN systems are presently in use: we monitor, study, control or manage different animal species, such as birds [67], zebras [46], bluefin tuna [7], and Antarctic fur seals [72]; environmental phenomena and disasters, such as volcanoes [34], glaciers [71], oceans [1], microclimate [17], avalanches [75], and fires [26]; human artifacts, such as buildings [65], bridges [49], smart homes [43], smart factories [6], parking lots [5], traffic jams [16], power grids [48], and cattle herds [11]; and detect, classify and track different objects for military purposes, such as tanks, naval vessels [86], enemy combatants [93, 62], and weapons [97].

The apparent success of WSNs in monitoring, management, tracking, and reacting to certain phenomena in the physical world is due to their fundamentally different capabilities compared to previous computational systems. A vast number of sensors can be embedded in the environment close to the phenomena of interest, over wide areas, yet distributed densely in space. Despite the large numbers of sensors, their small size allows them to be unobtrusive. Their autonomous operation and the ability to reconfigure provides WSNs with the ability to withstand failures of individual nodes and ways to adopt to unforseen environmental changes without human intervention. Thus, the main strength of WSN systems is that their power is not limited by the accuracy and quality of its individual components. Their true value is achieved through the collaboration and coordination of hundreds of sensors. For instance, a single sensor may only be able to detect the presence of a vehicle in its vicinity, whereas multiple sensors can collectively determine its speed, direction, and precise position.

Consequently, the design of WSN systems has become more challenging than that of the traditional distributed systems. Perhaps the most crucial design requirement is power efficiency, due to potentially long periods of time the sensors need to operate with limited power. This, along with the low per-sensor cost requirement, implies constraints on practically all hardware components, from microprocessors to memory units. Not only do the WSN applications need to be efficient in computation, memory usage, and network bandwidth, they also have to be fault tolerant. Failures of individual sensors due to hardware defects, environmental effects, or energy depletion, are expected to be the norm, rather than the exception. Also, it is expected that thousands of sensors may be deployed per application, possibly in an inhospitable environment and in an ad-hoc manner (e.g. thrown out from an airplane), making it infeasible to attend to individual sensors. WSN systems, therefore, need to be capable of autonomous operation, self-organization, and dynamic reconfiguration to react to the dynamic changes of environment or user requirements.

1.2 Time and Space

Coordination of sensors in time and space are critical requirements of many WSN systems. For example, in the application layer, structural or volcano monitoring need to correlate readings from multiple sensors to analyze the structural response of buildings or locate volcano eruptions. Vehicle tracking systems calculate an accurate position, direction, or speed of a vehicle from the timestamped distance estimates of individual nodes. Countersniper systems locate a shooter from the detection times of the shot at different sensors. In the systems layer, power efficiency is one of the most important requirements in WSNs. Since typically the most energy is spent on radio communication [2], it is important to reduce the size of data that needs to be communicated. Localized algorithms [24] are able to dramatically reduce the data size through coordination of multiple sensors. Further examples of system services that depend on the inter-node time or space coordination are the time division multiple access (TDMA) communication protocols that grant the access of multiple devices to a shared channel by assigning each device a unique transmission time-slot, debugging services that record execution traces along with the global time information, and geometric routing that achieves more efficient radio bandwidth utilization in multihop routing by utilizing the locations of the nodes.

Clearly, a common network-wide, or at least a neighborhood-wide view of the time and a consistent map are basic building blocks of many WSN systems. Services that establish the spatial and temporal reference systems are the *localization* and the *time synchronization* service, respectively. The same general design problems of WSN applications that we discussed before need to be addressed when designing localization and time synchronization services: limited energy resources, constrained hardware, dynamically changing environment, and the potentially large scale of the deployed WSN systems. Moreover, different WSN applications may have vastly different synchronization requirements, making it virtually impossible to design a single universal service. For example, acoustic source localization and beamforming applications require precise ground-truth sensor locations and timestamping of acoustic events with a few microseconds accuracy. On the other hand, tens of meters location accuracy of the sensors is sufficient in habitat monitoring, where the phenomena happen so infrequently, that the time of the delivery of the messages to a base station provides a sufficient time estimate. Debugging and logging services require a virtual global time service, whereas event based applications (such as acoustic source localization) require just momentary synchronization. Relative location and time information are sufficient for data aggregation, whereas some applications may require references to an external timescale or coordinate system. As a result, many of the well established time synchronization and localization algorithms are inadequate for majority of WSN applications and new protocols have been specifically developed for WSNs.

1.3 Contributions

In this thesis, we design, implement, and evaluate techniques that allow to coordinate time and space reference systems of a number of resource constrained sensor nodes utilizing their wireless communication capabilities.

In particular, we observe that existing time synchronization protocols work with only a limited number of hardware platforms and face significant problems in keeping up with hardware evolution. In fact, this is a general problem in the WSN domain: it is inconceivable to expect middleware developers to adapt their algorithms to all hardware platforms because this would require them to fully understand low level details of a variety of hardware. Similarly, the hardware manufacturers do not have capacities to implement the full spectrum of middleware services for their products. Especially so in the WSN domain, where the hardware typically becomes obsolete within a few years and its diversity is tremendous. We propose to address these problems by adapting a structured approach in the design of time synchronization protocols: we identify three layers of abstraction and define application programming interfaces (APIs) for the inter-layer communication. We argue that structuring time synchronization protocols in layers and standardizing interfaces of these layers improve adaptability, reusability, and portability of the protocols and help to decrease the complexity of their implementation. Further, we provide implementation, performance analysis, and quantitative comparison of a number of time synchronization services which will help the developers to identify the time synchronization needs of WSN applications and choose the most efficient service for their requirements.

Existing localization algorithms fail to deliver high accuracy and long range simultaneously while using low-cost hardware. We propose a novel radio interferometry based ranging algorithm that is superior to the existing algorithms in both the accuracy and the maximum range. Further, we study the Doppler effects in radio interferometric ranging to be able to compensate for the inaccuracies introduced by moving targets. Finally, based on the radio interferometric ranging, we implement cooperative localization and tracking services that achieve high accuracy and allow for low cost deployments.

This dissertation combines research ideas, algorithms, and experimental results contributed by a number of people. The generic framework for time synchronization protocols as well as the different time synchronization services were conceived during our collaboration with Prabal Dutta, Phil Levis, and David Culler from UC Berkeley. FTSP algorithm, developed by Miklos Maroti, was the first time synchronization algorithm to address node failures and dynamic topology changes while achieving very high accuracy. I contributed to FTSP by designing some of its failure resilience features and by extensive experimental testing. Next, I realized that for the purposes of reactive algorithms, such as acoustic source localization, high time synchronization accuracy can be achieved without proactive synchronization. Consequently, I enhanced a multi-hop routing protocol with time synchronization techniques to develop the resource efficient RITS algorithm. Akos Ledeczi pointed out that the RITS ideas can be used for centralized proactive time synchronization. I continued in this direction and developed the centralized RATS algorithm that can achieve ultra-low one hour duty cycling.

The basic building block of our localization and tracking algorithms is the radio interferometric ranging which has been conceived by Miklos Maroti. Miklos also provided the first implementation on the Mica2 mote that allowed us to create and observe the interference signal. I improved Miklos's initial ranging implementation to allow for large scale automated data collection. Utilizing the ranging algorithm, I performed a number of large scale experiments and analyzed the collected ranging data. This allowed us to improve range, scalability, and multipath resilience of interferometric ranging. Peter Volgyesi contributed by developing an algorithm to determine the phase and the frequency of low frequency interference signals on the constrained Mica2 hardware. The initial localization algorithm that allowed us to evaluate interferometric ranging used a genetic algorithm approach and was developed by Gyorgy Balogh. I realized that the localized node can be found at an intersection of hyperbolas if a number of anchor nodes are available to measure interferometric ranges. This allowed us to significantly improve the localization time. Consequently, me and Gyorgy Balogh designed and experimentally tested the initial tracking algorithm. The extension of the algorithm to track multiple nodes was developed by myself and Janos Sallai during our summer internship at Oak Ridge national laboratory. Consequently, I developed a way to significantly improve localization accuracy of mobile tracked nodes by utilizing Doppler effect. Finally, I developed the tracking algorithm that utilizes Doppler RF shifts only.

1.4 Organization

This thesis has two major parts: time synchronization presented in Chapter II and localization presented in Chapter III. Both sections are organized in a similar fashion: first we briefly summarize the related work, then we state the problem that we want to solve, design a solution, describe an implementation of the solution, and evaluate the implementation experimentally. We offer our conclusions and future work directions in Chapter IV.
Chapter II contains text and figures from the previously published papers:

- *The flooding time synchronization protocol* by M. Maróti, B. Kusý, G. Simon, and A. Lédeczi, in Proceedings of Conference on Embedded Networked Sensor Systems (SenSys), November, 2004.

- *On the scalability of routing-integrated time synchronization* by J. Sallai, B. Kusý, A. Lédeczi, and P. Dutta, in Proceedings of European Workshop on Wireless Sensor Networks (EWSN), February, 2006.

- *Elapsed time on arrival: a simple and versatile primitive for canonical time synchronization services* by B. Kusý, P. Dutta, P. Levis, M. Maróti, A. Lédeczi, and D. Culler, in the International Journal of Ad Hoc and Ubiquitous Computing, 2(1), 2006.

Chapter III contains text and figures from the previously published papers:

- *Radio interferometric geolocation* by M. Maróti, B. Kusý, G. Balogh, P. Völgyesi, A. Nádas, K. Molnár, S. Dóra, and A. Lédeczi, in Proceedings of Conference on Embedded Networked Sensor Systems (SenSys), November, 2005.
- *Node-density independent localization* by B. Kusý, G. Balogh, P. Völgyesi, J. Sallai, A. Nádas, A. Lédeczi, M. Maróti, and L. Meertens, in Proceedings of Conference on Information Processing in Sensor Networks (IPSN) SPOTS Track, April, 2006.
- *inTrack: High precision tracking of mobile sensor nodes* by B. Kusý, G. Balogh, A. Lédeczi, J. Sallai, and M. Maróti, in Proceedings of European Conference on Wireless Sensor Networks (EWSN), January, 2007.
- *Radio Interferometric Tracking of Mobile Wireless Nodes* by B. Kusy, J. Sallai, G. Balogh, A. Ledeczi, V. Protopopescu, J. Tolliver, F. DeNap, and M. Parang, in Proceedings of Conference on Mobile Systems, Applications, and Services (MobiSys), June, 2007.
- *Tracking mobile nodes using RF Doppler shifts* by B. Kusy, A. Ledeczi, and X. Koutsoukos, in Proceedings of Conference on Embedded Networked Sensor Systems (SenSys), November, 2007 (to appear).

Chapter II

Time Coordination

2.1 Background

Time coordination is important in WSN systems for two major reasons: a) to determine the temporal relationships of observations of an event by different sensors, and b) to coordinate actions of sensor nodes. For example, acoustic or seismic source localization systems [34, 93] deploy multiple spatially distributed sensor nodes in an area of interest and correlate the times of arrival of an acoustic or seismic event at different nodes to localize the event. Structural monitoring of buildings [49] or habitat monitoring [67, 46] systems generate a series of events and the data timestamps need to be consistent both within the series measured by a single node and across the series measured by multiple nodes. Virtually all applications that try to determine the relative locations or distances between the nodes require some kind of coordination in time to initiate the distance measurements, or to measure the time of flight of a signal [39]. Distributed logging or debugging services usually store data in the memory and analyze it later which requires the data to be timestamped using a common timescale.

The software component that is responsible for maintaining common timescale over a number of sensor nodes is called *time synchronization* service.

2.1.1 Time synchronization in traditional systems

While the clock accuracy and precision requirements are often stricter in WSNs than in traditional distributed systems, ironically, energy, computation, memory, and communication constraints limit the resources available to meet these goals. Time synchronization services need to be adaptive to tolerate link and node failures which further complicates their design.

These challenges prevent the direct application of traditional time synchronization techniques in WSNs. For example, delivery order of network messages in WSNs does not assure the causality of events and therefore, Lamport's clock [59] approach is not applicable. Free running clock crystals, rather than the message order, determine the causality of events in WSNs. Another protocol commonly used in traditional distributed systems is the Network Time Protocol (NTP) [76, 77]. However, NTP assumes that the CPU and the network are always available while WSN systems frequently power down to prolong their lifetime. Also, NTP assumes that node-to-node communication latency is bound, while in WSNs, wireless communication over multiple hops, or even over a single hop in high contention situations, exhibits a significant variance in message delivery time. Therefore, time synchronization protocols designed for traditional distributed systems are rarely used in the WSN domain.

2.1.2 Types of synchronization

Different WSN applications have different time synchronization needs which can typically be implemented with different levels of complexity, incurred overhead, and accuracy. One way of addressing the time synchronization problem is to implement a single all-purpose algorithm that satisfies even the most stringent requirements. However, much better resource utilization is possible when using a time synchronization service that is specifically tailored to the needs of a particular application. Therefore, it is important to study time synchronization requirements of WSN applications which can be done along multiple dimensions:

- **Accuracy:** At one end of the spectrum is habitat monitoring which monitors phenomena at a slow enough time scale that delivery order of the radio messages at a base station provides reasonable accuracy. More stringent is structural health monitoring that requires timestamping to within a fraction of a packet transmission time. At the other end of the spectrum are acoustic source localization or beamforming applications that require signal phase be measured to within just a few tens of microseconds.

- **Local-Neighborhood versus Network-Wide:** Some applications or services may require just in-neighborhood consistency of local times. For example, power management services schedule the radio communication between multiple nodes at particular time instances allowing the nodes to switch of their radios in between. On contrary, many applications require network-wide consistency with an externally provided timescale such as UTC.

- **Continuous versus Reactive:** Continuous (proactive) mode is a popular way to implement time synchronization in WSNs. Each sensor node is required to be synchronized at all times which consumes a lot of power and communication resources. However, due to the conceptual simplicity of this service, many WSN application developers resort to it without even considering other approaches. Although the use of proactive synchronization can be justified in some cases, such as distributed logging, most of the applications incur the extra overhead unnecessarily. Alternatively, synchronization can be achieved on-demand, in a so called ”reactive mode”, where the common timescale is determined after the event of interest has happened. The reactive synchronization allows for both high accuracy and efficient resource utilization at the same time [88, 93].

2.1.3 Clock sources

Although different types of electronic circuits are available to drive physical clocks, for the purposes of WSN applications, common quartz crystals provide a reasonably accurate, small, energy-efficient, and inexpensive solution. A quartz crystal is an electronic circuit that converts the mechanical resonance of a vibrating crystal to a digital signal with a precise frequency. The oscillations of the crystal are usually counted in a clock-tick counter. Each sensor node maintains its own local clock utilizing a quartz crystal. The frequency at which a given crystal oscillates defines the period of the physical clock, or granularity of the timescale. Two important properties of the crystals are frequency accuracy, defining how close to a nominal frequency the crystal oscillates, and frequency stability, defining how much the frequency of the crystal changes over time. Both these properties are important design parameters for time synchronization services because they specify the deviations of the clock from its expected values.

Clock errors and their compensation: In general, clock-tick counters have different values at different sensor nodes and therefore they need to be synchronized. The difference in time shown by the clocks at the different nodes is called *clock skew*. Clock skews can be compensated for either by clock-tick counter adjustment, or by finding a function transforming the clock value at one node to the timescale of the other node. The following clock errors influence the accuracy of synchronization:

- **Time Representation Error**: this error corresponds to the delay between an event and the nearest (or next) representable time value.
- **Clock Drift:** refers to the phenomena where the difference of the clock frequencies at different devices causes the clock skew to change over time. Clock drifts are caused by environmental factors and slight deviations of each clock circuitry from its specifications.

The frequency stability of the clocks is an important parameter of the clock drift estimation algorithms. Several factors can cause permanent or temporary deviations of the clocks from their nominal frequency:

- **Long Term Effects**: manufacturing imprecision and some environmental factors (e.g. mechanical shock) result in permanent frequency errors of crystals.
- **Short Term Effects:** ambient temperature, air pressure, or humidity can result in temporary frequency errors of the crystals. Relatively frequent recalibration of the clocks is required to minimize short term clock errors. Alternatively, more advanced

circuits, such as temperature controlled oscillators, can be used to compensate for changing environmental conditions.

Typical drift rates of commonly used clocks (i.e., quartz crystals) are 50 parts per million (ppm), thus the clock skews can change by a few tens of μs per second. This would mandate continuous adjustment of the clock skew with a period of less than one second to keep the error in the μs range, which would strain the system resources. Alternatively, if the clocks have a good short-term stability, their skew can be estimated with statistical techniques. Due to its accuracy and simplicity of implementation, the most commonly used technique is *linear regression* [20].

2.1.4 Synchronizing by the radio

The most popular technique to synchronize local clocks of different nodes is timestamping the transmission and reception of a radio message. The main reasons are the wide availability of radio chips at sensor nodes, modest power requirements of the radio transmission, and negligible propagation delays due to the speed-of-light propagation of radio signals. Two dominating radio timestamping techniques are:

- **Receiver-Receiver Synchronization (Reference Broadcast)** [20]: a dedicated beacon node broadcasts a radio message that is received by multiple receivers. The receivers exchange the time of arrival of the beacon message, thereby synchronizing their clocks. The main advantage of this method is that the message does not contain any time synchronization error caused by the sender, nor is it important when the message was sent. However, an extra communication step is required to exchange the timestamps between the receivers.

- **Sender-Receiver Synchronization** [28]: this method was designed to synchronize a sensor node to a global time server. The synchronizing node transmits a request message and waits for the response from the global time server. The response message contains two timestamps recorded in the global timescale: the time of reception of the request message and the time of transmission of the response message. The synchronizing node can then compute the skew of its clock with respect to the global clock. However, the communication is carried over unicast links which means that the communication overhead increases linearly with the number of synchronizing nodes.

- **Hybrid One-Way/Two-Way Synchronization** [44]: a combination of one-way and round-trip synchronization has been also suggested to overcome certain limits of the round-trip synchronization. It was shown to improve clock skew estimation, however, the hybrid method is still susceptible to the same communication related problems as the round-trip synchronization.

Multihop synchronization: Typically, WSN systems are deployed over areas much larger than the communication range of the individual sensors. Therefore, time synchronization algorithms need to be extended over multiple hops by disseminating timing information from points in network that have access to synchronized clocks. Several approaches to multi-hop synchronization have been described:

- **Multiple Global Time Sources**: the problem of multi-hop synchronization is avoided by deploying a large number of devices that have access to the global clock by some external mechanism (such as GPS). Each node needs to have a direct link to at least one global time source.

- **Weighted Shortest Path Search** [20]: the sensors and the communication links are represented as a graph, with edges weighted based on the quality of the time conversion along that edge. Multi-hop synchronization of two nodes is performed as a series of single hop synchronization along the weighted shortest path between the two nodes.

- **Tree Construction** [33]: a spanning tree is constructed in the network, and a single hop synchronization is performed along the edges of the tree. The disadvantage is that it is hard to maintain a tree structure in a dynamically changing topology with asymmetric links, the synchronization error grows with the increased distance from the root of the tree, and nodes with a large degree need to serve multiple receivers, depleting their energy resources.

Message timestamping errors: Careful analysis and compensation of the sources of error in message timestamping is one of the most important steps to achieve highly accurate time synchronization. The delay in the message delivery is not a problem in itself, if it is deterministic. The challenge is to compensate for the nondeterministic delays which, unfortunately, can be magnitudes larger than the required accuracy of synchronization. The following decomposition of the message latency errors was introduced by Kopetz [50]:

- **Send Time**: the time it takes from the request to send a radio message until the MAC layer on the transmitter side receives the command. Kernel processing, message assembly and often system calls need to be executed in the process, so this delay depends on the system call overhead and the processor load.

- **Access Time**: the delay incurred waiting for access to the communication channel until the actual transmission begins. This delay depends on the channel contention and is highly nondeterministic.
- **Propagation Time**: the time it takes for the message to travel from the sender to the receiver. This depends only on the distance between the two nodes and is typically negligible compared to other delays.
- **Receive Time**: time to process the incoming message and to notify the receiver application. It depends on factors similar to the send time.

Typically, random access MAC protocols are used in WSNs which lead to unpredictable and arbitrarily long delays in message transmission times under high contention. It has been observed [20] that the propagation time is close to zero and the receive time introduces delays which are deterministic. Therefore, receiver-receiver synchronization that avoids the send time and the access time errors, provides accurate results. Further study of the timestamping errors was done by Ganeriwal et al. [28], extending the former error characterization by

- **Transmission Time**: the time required for the sender to transmit the message. This depends on the length of the message and the speed of the transmission.
- **Reception Time**: the time required for the receiver to receive the message. The transmission and reception times have similar characteristics and overlap, if the propagation time is negligible.

Operating systems for WSNs, such as TinyOS [41], provide software hooks to access low-level hardware drivers. Consequently, timestamping of the radio messages can be delayed until the last moment. In fact, the message can be timestamped in the MAC layer, after the node has acquired the access to the radio channel, eliminating the send and the access times completely. Therefore, the sender-receiver synchronization can achieve similar accuracy as the receiver-receiver synchronization, without incurring the extra communication overhead required for reference broadcasting [28].

2.2 Related Work

A number of time synchronization protocols for WSNs can be found in the literature. We present an overview of the most important algorithms in this section.

- **Time-Stamp Synchronization (TSS) [88]** is a post-facto, reactive time synchronization algorithm. Synchronization in TSS is only performed at message transmission, when the times are transformed from the timescale of the sender to that of the receiver. This transformation is done by computing the age of the timestamp from when the event was registered until the timestamp was received. One drawback to the TSS approach is that it utilizes unicast communication for estimating the message delivery delay. The accuracy of this method was shown to be 200 μs per hop on a PC class device and the communication cost is twice the length of the multi-hop route between the source and the target node.

- **Reference Broadcast Synchronization (RBS) [21]** perhaps the best known time synchronization algorithm in the sensor network regime, uses a transmitter node to synchronize the clocks of two receivers to each other. The RBS approach uses receiver-receiver timestamping approach which eliminates the delays on the sender side, most notably in the MAC layer. The accuracy of the RBS timestamping reported by the authors is approximately 11 μs on the Mica [40] platform. In the multi-hop scenario, RBS constructs a multi-hop chain of gateways between any two nodes that translate the times between the distant nodes. This chain of nodes can be found as a result of a weighted-shortest-path search. A disadvantage of this approach is the overhead and complexity caused by the broadcasts exchanged between neighbors to achieve pair-wise synchronization.

- **Lightweight Time Synchronization (LTS) [33]** focuses on reducing the communication and computation requirements of synchronization, rather than achieving the maximum possible accuracy. The algorithm provides a way to adjust efficiency (the resynchronization period) based on the specified accuracy and the known drifts of the clocks. Two ways to synchronize nodes in the network are presented: a spanning tree based push method and a distributed pull based method. Its disadvantage is the assumption of the existence of externally synchronized nodes which can become bottlenecks and critical points of failure in both variants of the algorithm.

- **Timing-Sync Protocol for Sensor Networks (TPSN) [28]** eliminates sources of timestamping errors by making use of the implicit acknowledgments to transmit information back to the sender. This protocol gains an additional accuracy over RBS due to timestamping the radio message twice and averaging these timestamps. The protocol is divided into two different phases. In the level discovery phase, each node is assigned a level which reflects the hop count of the node to the root node. In the synchronization phase, every node exchanges timestamps with its parent. Since TPSN relies on a sender-receiver information exchange, messages cannot be broadcasted which results in higher communication load. The TPSN algorithm was implemented on the Mica platform, resulting in an average error of 16.9 μs in a single hop case.

2.3 Problem Statement

Time synchronization in WSNs was studied extensively in the past decade. We believe that there are two major problems in how the existing time synchronization algorithms are designed, implemented and used in WSN applications.

First, we observe that time synchronization algorithms are typically designed and evaluated at different hardware platforms which makes their direct comparison a tedious task. For example, TSS [88] was evaluated on PCs, RBS [20] on IPAQs, and TPSN [28] on Berkeley Mica motes. Moreover, these algorithms are designed in an ad hoc way, resulting in their limited portability across different hardware platforms.

Second, WSN applications require different levels of accuracy, power efficiency, or resilience to hardware failures and a variety of time synchronization algorithms were developed to support these different requirements. However, it is difficult for WSN application developers to abstract common usage patterns from a number of existing time synchronization algorithms and determine the type of synchronization that is the most suitable for their needs. In practice, developers end up using the virtual global time service due to its conceptual simplicity. The virtual global time service is typically implemented with proactive synchronization techniques which may waste precious system resources unnecessarily.

We propose to address these problems by defining layers of abstraction in the time synchronization protocol stack and specifying application programming interfaces (APIs) for the inter-layer communication. We argue that structuring time synchronization protocols in layers and standardizing interfaces of these layers improves adaptability, reusability, and portability of the protocols and helps to decrease the complexity of their implementation. Further, we provide implementation, performance analysis, and quantitative comparison of a number of time synchronization services which will help the developers to identify the time synchronization needs of their applications.

2.4 Organization

First, we briefly describe the proposed layers of abstraction in Section II.5. We revisit the sources of radio timestamping error in Section II.6 which helps us design the hardware abstraction layer. Then we describe, implement, and evaluate a universal timestamping primitive in Section II.7. Finally, we describe, implement, and evaluate three time synchronization services in Sections II.8,II.9, and II.10.

2.5 Layers of Abstraction in Time Synchronization

Current operating systems [18, 41] and network protocols commonly organize a set of functionalities in layers, presenting the clients of these layers with an abstract view of the provided functionality. The layered approach improves reusability and portability of software over different hardware platforms by isolating the required code changes to a single layer and simplifies the development of applications by allowing them to work on a more abstract, hardware-independent level. However, hardware abstraction is often associated with performance degradation of the applications because we loose the flexibility to fine-tune their implementation [18]. Therefore, it is important how the layers of abstraction of time synchronization protocols are structured. On one hand, the abstraction needs to be specific enough at a certain level, to allow for its efficient implementation on different hardware platforms. On the other hand, time synchronization protocols have to provide more abstract hardware-independent interfaces for the purposes of WSN applications.

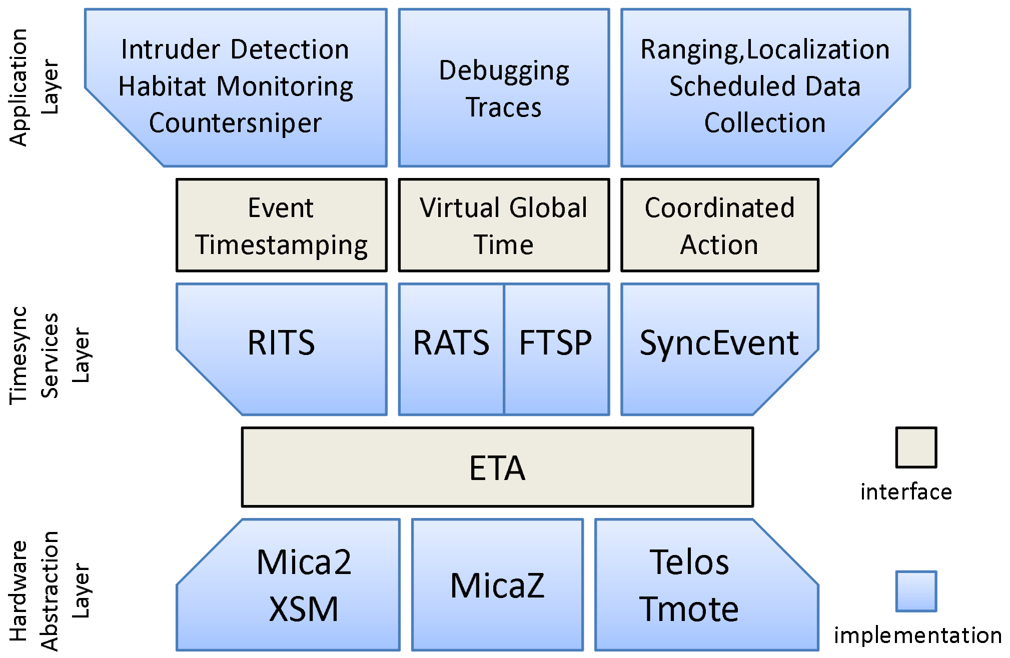

Figure II.1: Layers of abstraction in time synchronization protocols.

We propose a three-layer architecture of time synchronization protocols, shown in Fig. II.1. The hardware abstraction layer improves the adaptability, reusability, and portability of the protocols and allows for fine-tuned performance and efficiency of their implementation. The time synchronization services layer implements the actual time synchronization protocols, each of them providing different levels of accuracy, power consumption, and fault tolerance. Application layer algorithms use the time synchronization services through standardized interfaces which allow them to utilize these services on a conceptual level.

2.5.1 The hardware abstraction layer

We build the time synchronization protocol stack based on a simplified version of sender-receiver timestamping, the *Elapsed Time on Arrival (ETA)* primitive. Other techniques could be adopted as a basic time synchronization primitive, for example the receiver-receiver or sender-receiver radio timestamping techniques (Section II.1.4), or even timestamping the arrival of an acoustic signal instead of the radio signal. However, we will show that in the WSN domain, ETA allows for high accuracy of synchronization as well as efficient utilization of system resources, thus overcoming the deficiencies of other techniques.

2.5.2 The time synchronization services layer

The proposed ETA primitive would not be very useful, if it only supported efficient implementation of a limited number of time synchronization services, despite its demonstrated portability over numerous hardware platforms. We show that ETA allows for implementation of a number of time synchronization services. Although we do not claim that this set of services is complete, we expect that ETA could be used to implement other services as well.

In particular, we develop and evaluate three different services and quantify their strengths and weaknesses against each other. The event timestamping service allows to timestamp observations of an isolated event by different nodes and to correlate the timestamps at a sink node using its own local time. The virtual global time service provides a single, virtual, and globally shared timebase throughout the network. Finally, the coordinated action service allows multiple nodes to agree on a single point in time in the future. All services are implemented using ETA, allowing for high time synchronization accuracy and portability of the services to several hardware platforms.

2.6 Revisiting sources of time synchronization errors

The sources of error in message timestamping need to be carefully analyzed and compensated for because non-deterministic delays in radio message delivery can be magnitudes larger than the required time synchronization accuracy. In Section II.1.4, we have introduced the decomposition of these errors to *send time*, *access time*, *transmission time*, *propagation time*, *reception time*, and *receive time* (see Fig. II.2).

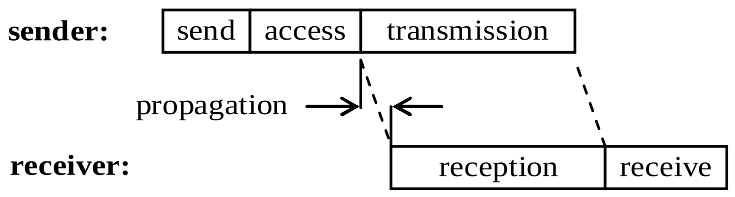

Figure II.2: Basic decomposition of message timestamping errors.

We further analyze the sources of uncertainties in the overlapping transmission and reception times by observing an idealized point of radio message, such as the end of a particular byte. We follow the transmission of this idealized point through the software, hardware and physical layers of the wireless channel from the sender to the receiver. The following delays in the propagation of the idealized point seem to be the most important:

(1) *Interrupt Handling Time*—the delay between the radio chip raising and the CPU responding to an interrupt by recording a timestamp. This time is mostly less than a few microseconds (waiting for the CPU to finish the currently executed instruction), however when interrupts are disabled in atomic sections this delay can grow large.

(2) *Encoding Time*—the time it takes for the radio chip to encode and transform a part of the message to electromagnetic waves starting from the point when it raised an interrupt indicating the reception of the idealized point from the CPU. This time is deterministic and is in the order of a hundred microseconds.

(3) *Decoding Time*—the time it takes for the radio chip on the receiver side to transform and decode the message from electromagnetic waves to binary data. It ends when the radio chip raises an interrupt indicating the reception of the idealized point. This time is mostly deterministic and is in the order of hundred microseconds. However, signal strength fluctuations and bit synchronization errors can introduce jitter.

Some radio chips cannot capture the byte alignment of the transmitted message stream on the receiver side and the radio stack has to determine the bit offset of the message from the alignment of a known synchronization byte. Since the transmission time of a byte can be a few hundred microseconds, the delay caused by the incorrect byte alignment must be compensated for.

(4) *Byte Alignment Error*—the delay incurred because of the different byte or data segment alignment of the sender and receiver. This is deterministic and can be computed at the receiver from the bit offset and the speed of the radio.

We have measured these timestamping errors on Mica2 platform: the interrupt handling time is typically around $5\mu s$ depending on the length of the code path between the start of interrupt handler and the part that records the local time. However, we observed interrupt handling times as high as $30\mu s$. The sum of encoding and decoding times is between $110\mu s$ and $112\mu s$. The byte alignment time is between $0\mu s$ (for bit offset 0) and $365\mu s$ (for bit offset 7). In contrast, the propagation time is under $1\mu s$ for distances under 300m.

2.6.1 Eliminating timestamping errors

The uncertainties of the send, receive, access and byte alignment times are best eliminated by timestamping the message in the MAC-layer, as is done in several time synchronization protocols. The propagation time cannot be calculated or compensated for within the transmission of a single message. However, in static networks it has no jitter and over short distances, its duration is negligible. We argue that the message needs to be timestamped at an idealized point, practically when an interrupt is raised by the radio. This eliminates the effect of the overlapping transmission and reception times. The jitter in the interrupt handling time is best eliminated by utilizing a capture register on the CPU. If the timestamp is taken in software, it is important to minimize the length of all atomic sections in the application to minimize the amount of time interrupts are disabled. The encoding time lasts several hundreds of microseconds but usually has very low jitter. On the other hand, the radio technology, signal strength fluctuations and bit synchronization errors will introduce jitter in the decoding time.

The presented technique allows only to reduce the jitter of these transmit and receive timestamps. However, the actual absolute value of the the difference between the sender and receiver side timestamps can be measured with a one-time calibration procedure.

2.7 The Elapsed Time on Arrival Timestamping Primitive

As discussed before, the hardware abstraction layer needs to be designed carefully to support both its accurate and efficient implementation on multiple hardware platforms and to enable implementation of multiple time synchronization services. We have designed a new radio timestamping primitive, Elapsed Time on Arrival (ETA), to be the point of abstraction between the low-level operating system and time synchronization services. ETA is a simplified version of sender-receiver timestamping and works as follows: a sender node transmits a single radio message and all neighboring nodes establish a common time base with the sender after receiving the mssage. The key idea is that the sender includes the time of the message transmission as a separate field in the message. All receivers of that message can then synchronize their local clocks to the clock base of the sender by timestamping the arrival of the message.

The advantage of ETA over other techniques found in literature is its simplicity—a single radio message can synchronize all nodes in the radio range of the sender. In contrast, receiver-receiver synchronization incurs extra communication overhead to exchange arrival times of the reference broadcast between the synchronizing nodes. Sender-receiver synchronization requires point-to-point communication, rather than broadcast, which results in higher communication overhead. However, it is important to note that it would be difficult to implement ETA on modern operating systems because it requires low-level access to the radio driver to be able to modify radio messages that are being transmitted. Receiver-receiver synchronization would be a better choice in this case. TinyOS is an open-source, modular operating system specially designed for WSNs that allows such low-level access through software hooks. To summarize, ETA requires the following functionality:

```
Radio.OnBeginTransmission(TimeStampedMessage msg);
Radio.OnBeginReception(TimeStampedMessage msg);
```

where `Radio.OnBeginTransmission` is a function called at the sender after the access to the radio channel has been granted and before the first byte of the message is transmitted. `Radio.OnBeginReception` is a function called at the receiver when the first byte of the message is received and it allows the receiver to record the time when the message arrives. Even though `Radio.OnBeginTransmission` and `Radio.OnBeginReception` are called by different radio chips, they should correspond to time instants separated by a measurable, low-variance time interval. Further, ETA needs to extend radio messages to contain a field storing the transmission time.

Our goals were to achieve high time synchronization accuracy and efficient use of resources. The efficiency of the ETA primitive comes from its simplicity: a single message can synchronize all nodes located in the radio range of the sender. In fact, synchronization with virtually no overhead can be achieved by piggy-backing the ETA timestamp on regular radio traffic. To achieve high accuracy, we needed to understand possible sources of error in detail as we describe in Section II.6.

2.7.1 ETA implementation

For time synchronization services, it is more helpful to consider the time when a certain phenomenon of interest, or *an event*, was detected, rather than when a certain time synchronization message was transmitted. Therefore, ETA implementation performs the following inter-node conversions of the event time: at message transmission, the elapsed time since the event occurrence is computed and recorded in the radio message. On receipt, a node computes the event time in its timebase by subtracting the elapsed time from the arrival time of the message.

Program 1 The Elapsed Time on Arrival (ETA) primitive.

```
struct TimeStampedMessage {
    char[] payload;
    int eventTime;
    int transmissionTime; // not actually transmitted
    int receptionTime;    // not actually transmitted
}

OnBeginTransmission(TimeStampedMessage msg) {
    msg.transmissionTime = GetLocalTime();
    msg.eventTime = msg.transmissionTime - msg.eventTime;
}

OnBeginReception(TimeStampedMessage msg) {
    msg.receptionTime = GetLocalTime();
}

OnEndReception(TimeStampedMessage msg) {
    msg.eventTime = msg.receptionTime - msg.eventTime;
}
```

The pseudo code of ETA is shown in Program 1: each node accesses its local clock or timer through the `GetLocalTime` call. On an event occurrence, the local time is recorded in the `eventTime` field and the event message is enqueued for the transmission. When the event message is scheduled and the idealized point of the message is being transmitted (ostensibly after some delay in the media access control layer), both the sender and the receivers record their current local time into the `transmissionTime` and the `receptionTime` fields, respectively. The sender also updates the `eventTime` field to contain the elapsed time since the event occurrence which happens before the actual bytes of the `eventTime` field are transmitted. Note that all time-related information is contained in the `eventTime` field, thus the `transmissionTime` and `receptionTime` fields are not transmitted over the wireless channel to the neighbors. After the whole message is received, `eventTime` is reconstructed on the receiver side by subtracting the elapsed time from the `receptionTime`. The key enabler of ETA is the ability to update the content of the radio message (to update the `eventTime` field) while the message is being transmitted.

The proposed timestamping protocol eliminates or reduces all sources of timestamping error except for propagation time and the clock drift between the sender and receiver while a message is waiting in the queue. Neither of these errors can be estimated with a single radio message and hence, this estimation is left for higher layers to implement if necessary. The overhead of the algorithm is a single field in the transmitted message. After the message transmission is complete, all receivers know the time of the event in their local time base.

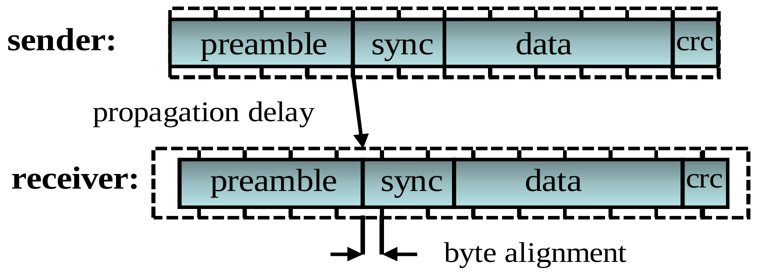

Figure II.3: Message transmission with Chipcon CC1000 radio chip and Mica2 mote.

Mica2 and ExScal mote implementation: The Mica2 and ExScal hardware platforms [40, 19] both use Chipcon's CC1000 radio chip [12], a byte oriented radio. The message transmission and reception proceed as follows (see Fig. II.3): a message broadcast starts with the transmission of preamble bytes, followed by SYNC bytes, then with the actual message data bytes, and ends with CRC bytes. During the transmission of the preamble bytes the radio of the receiver synchronizes to the carrier frequency of the incoming signal. From the SYNC bytes the receiver calculates the bit offset to be able to reassemble the message with the correct byte alignment. The CRC bytes are used to verify that the message was not corrupted.

ETA effectively reduces the jitter of the interrupt handling and encoding/decoding times by recording multiple timestamps both on the sender and receiver sides. The timestamps are recorded at each byte boundary after the SYNC bytes as they are transmitted or received. First, these timestamps are normalized by subtracting an appropriate integer multiple of the nominal byte transmission time, the time it takes to transmit a byte. The jitter of interrupt handling time is mainly due to program sections disabling interrupts on the CPU for short amounts of time. This error is not Gaussian, but can be eliminated with high probability by taking the minimum of the normalized timestamps. The jitter of encoding and decoding time can be reduced by taking the average of these interrupt error corrected normalized timestamps. On the receiver side, the final averaged timestamp must be further corrected by the byte alignment time that can be computed from the transmission speed and the bit offset. Note that, even though multiple timestamps are recorded, only the final error corrected timestamp is transmitted.

The number of transmitted bytes puts an upper limit on the achievable error correction using this technique. However, with only 6 timestamps, the mean timestamping precision can be improved from tens of microseconds to $1.4\mu s$ on the Mica2 platform. Since a TinyOS message contains a mandatory 6 byte header, it is always possible to calculate the error-corrected timestamp.

We evaluated ETA in the following experiment: 4 motes were sending timestamped messages to each other for 10 minutes, each with a 5-second sending period. The timestamps were recorded both on the sender and receiver sides, and the pairwise clock skews of the nodes were determined offline with linear regression. The timestamping error is the absolute value of the difference of the receiver timestamp and the linearly corrected sender timestamp. The average and maximum timestamping errors were $1.4\mu s$ and $4.2\mu s$, respectively.

MicaZ and Telos implementation: Both Micaz [74] and Telos [83] platforms utilize Chipcon CC2420 radios [13], which supports the IEEE 802.15.4 standard. CC2420 belongs to the latest generation of radios that make the processor-radio interface richer without burdening the processor with the overhead of directly coordinating the transmission. Such improvements also enable fine-grained timing information to be communicated from the radio to the processor. In particular, similarly to the ethernet frame, CC2420 utilizes an 8-bit Start of Frame Delimiter (SFD)[1] and provides a deterministic hardware interrupt on transmission/reception of the SFD byte. CC2420 also provides random access to the transmit FIFO *during* transmission which together with the SFD interrupt enable fine-grained message timestamping with virtually no jitter on the transmitter end. Accurate timestamping on the receiver end is possible by connecting the SFD signal to a timer capture pin on the processor. Both our Micaz and Telos implementations utilize the SFD interrupt capture to achieve accuracy on the order of the time representation error. Since the two platforms use a different CPU, the local clocks have different granularity. Inter-operability of time synchronization of MicaZ and Telos motes can be achieved by converting the values of the local clocks to a common timeframe.

2.8 Time Synchronization Services

Many WSN applications require time synchronization but the *model* of synchronization that is needed varies greatly. In practice, many application developers resort to the virtual global time service without considering other alternatives due to its conceptual simplicity.

[1]A bit pattern that marks the start of the actual radio message frame.

Table II.1: Time synchronization services, their typical applications, and proposed API calls and events that can be used in a general implementation. Events are callbacks from a service layer to a higher layer and are of the form `OnXXX`.

Canonical Service	**Typical Applications**	**APIs**

Event timestamping	Intrusion detection
	Countersniper system
	Source localization
	Habitat monitoring
	Volcano event monitoring
		`SendToSink`
		`SendToAll`
		`OnReceive`

Virtual global time	Debugging traces	`GetGlobalTime`
		`LocalToGlobal`
		`GlobalToLocal`

Coordinated action	Scheduled data collection Communications scheduling Ranging and Localization	`Schedule` `OnSchedule`

However, virtual global time requires proactive, message-intensive operation, thus using large amount of system resources. We argue that abstracting common time synchronization usage patterns from a number of existing applications and standardizing the interfaces of these abstractions will help developers better identify the synchronization needs of sensor network applications and help them choose the most efficient model for their needs. Table II.1 summarizes the services, their APIs, and typical WSN applications.

2.8.1 Event timestamping service

A single, isolated event is observed and timestamped by one or more nodes at possibly different times. The observation from each node is to be collected at one or more sink nodes, either individually or in aggregate form. The sink node must determine the timestamp of each event in the sink's local time, *regardless of whether the sink is coordinated with an external source of time such as GPS*. This service extends to multiple events through repeated invocation.

An event timestamping API is shown in Program 2. The `Packet` data structure is augmented with an `eventTime` field, similarly to ETA's `TimeStampedMessage` in Program 1. The API defines: the `SendToSink` call to convergecast a packet (i.e. send a packet toward the sink node), the `SendToAll` call to flood a packet to all other nodes, and the `OnReceive` event to signal when a new packet is available at a node (signaled at every intermediate hop). When `OnReceive` is signalled, the `eventTime` data field of `Packet` contains the time of event in the local time of the current node.

Program 2 Event timestamping service interface.

```
    command bool SendToSink(Packet packet);
    command bool SendToAll(Packet packet);
    event void OnReceive(Packet packet);
```

2.8.2 Virtual global time service

Virtual global time is perhaps conceptually the simplest of the time synchronization services to understand but is often the one which requires the most overhead. The basic idea is straightforward: a single, virtual, and globally shared timebase is accessible at each node in the network. An important question is where the timebase originates. Options include an external source such as GPS, a distinguished root node, or some kind of network statistic such as the network mean.

The virtual global time API (Program 3) includes calls to read the current global time (`GetGlobalTime`) and to convert between the local times and their corresponding global times (`LocalToGlobal,GlobalToLocal`).

Program 3 Virtual global time service interface.

```
  command bool GetGlobalTime(int time);
  command bool LocalToGlobal(int time);
  command bool GlobalToLocal(int time);
```

2.8.3 Coordinated action service

Coordinated action is similar to event timestamping, except that the event is in the future. Applications that require coordinate action include scheduled data collection and structural health monitoring among others. In the former, the action is discrete data collection. Depending on the different required fidelities, the synchronization needed for coordinated action can be used for the events themselves. In the latter, the action is continuous data collection.

The challenge in coordinated action is that, unlike simple timestamping, it cannot benefit from post-facto synchronization such as an RBS exchange. However, unlike virtual global time, the nodes do not need a continuous synchronized time value: they merely need to agree on a single point in time. While an implementation of this service can depend on virtual global time, other implementation options exist.

We propose a coordinated action API (Program 4) that includes calls to schedule an action in a node's *local* time (`Schedule`) and signal an event when it is time to act on some previously received coordinated action request (`OnSchedule`).

Program 4 Coordinated action service interface.

```
command bool Schedule(int time);
event void OnSchedule();
```

2.9 Implementation of Time Synchronization Services

Often, different WSN applications require the same time synchronization service, but make different assumptions about the network topology, the expected lifetime of the network, the cost of the deployed hardware, etc. In such cases, more than one implementation of the time synchronization service should exist to offer an optimal solution for the different application scenarios. For example, a reliable and GPS-equipped base station with wired power node may be part of the deployment setup. Such a node can maintain and disseminate global time information while keeping the risk of the global time service failure minimal. On the other hand, if sensors are deployed randomly in a remote area, a robust reconfiguration mechanism should be provided in case the global time source fails.

We implement time synchronization services described in the previous section and compare their performance under similar experimental setups. Comprehensive comparisons of this kind are important because they can guide other developers in making informed decisions about time synchronization protocols.

2.9.1 Routing integrated time synchronization (RITS)

Commonly, proactive protocols use periodic message broadcasting to compensate for the clock skews of the sensors. The need for periodic message exchange, however, conflicts with power constraints and lifetime requirements of WSN applications. The observation that the global virtual time of an event is not used at the node registering the event, but only at the data fusion node, lead to the development of resource-efficient *reactive time synchronization protocols* [88, 21]. The general idea of the reactive approach is to maintain the age of an event as opposed to the global time. Upon the arrival of the packet containing the event information, the data fusion node can compute the event time in its local timescale from the age of the event and the message arrival time. Synchronization takes place after the event occurred; henceforth, this approach is often called post-facto synchronization.

We propose the reactive ***Routing Integrated Time Synchronization*** protocol to implement the ***Event Timestamping Service***. It can be used to obtain microsecond-accurate times of an event detected at a number of spatially distributed nodes in a common timebase. RITS is a straightforward extension of ETA over multiple hops using the multi-hop routing engine DFRF [68].

On detecting an event, the application on the sensor node creates a data packet containing the event information which includes the local time of the event detection. The packet is consequently handed over to the routing service, which delivers it to the sink. RITS places no assumptions on the network topology or routing algorithm beyond those that are required by the application. Rather than performing explicit time synchronization after the event of interest is detected which is how the traditional post-facto time synchronization protocols work, RITS performs inter-node time translation along the routing path from an observer node to the sink: as the data packet travels from node to node in the network, RITS converts the corresponding timestamp from the local time of the sender to that of the recipient. When the packet arrives at the sink, the routing service provides the application layer with the event timestamp, converted to the local time of the sink. Hence, event observations at multiple nodes can be correlated since they use a common timebase.

The RITS implementation is shown in Program 5. Upon receiving a request to send a data packet, RITS creates a `TimeStampedRITSMessage` and utilizes ETA to perform inter-node time conversions. Upon receiving a `TimeStampedRITSMessage`, the node creates a new packet and associates with it `message.eventTime` which is the local time of the event. RITS then signals the reception of packet to the application level and possibly enqueues the packet for the further transmission.

2.9.2 Centralized proactive time synchronization (RATS)

Rapid Time Synchronization, a proactive time synchronization protocol, is an implementation of the ***Virtual Global Time Service*** for medium-sized networks providing microsecond time synchronization accuracy and rapid convergence. The basic idea is the same as in RITS, except that the direction of messages is reversed: instead of convergecast from multiple nodes to the sink, we use a broadcast from a single node, the root, to all other nodes in the network. The event time that the root disseminates is its current local time. The root node is selected prior to the deployment and thus should be resilient to hardware failures as no other node in the network will be able to become the root. In many deploy-

Program 5 Pseudo code of the RITS implementation.

```
struct RITS_Packet {
    char[] data;
    int    eventTime;
}

SendToSink(RITS_Packet packet) {
    TimeStampedRITSMessage msg;
    msg.payload  = packet.data;
    msg.eventTime = packet.eventTime;
    MessageSend(msg);    // transmit over the radio
}

OnMessageReceive(TimeStampedRITSMessage msg) {
    RITS_Packet packet;
    packet.data = msg.payload;
    packet.eventTime = msg.eventTime;
    signal OnReceive(packet);
    Enqueue(packet);    // transmit via routing
}
```

ments, sensor networks depend on typically a single gateway node that connects the sensor network to a PC-class device and hence it is a good choice for the root.

The root initiates time synchronization by broadcasting a message containing `seqNumber` initialized to 0, `rootEventTime`, and `eventTime` initialized to the same value (the current time of the root at transmission). Network-wide broadcast is used to disseminate the root's message to all nodes in the network. The `seqNumber` and `rootEventTime` fields are not modified during the broadcast, while ETA converts the `eventTime` field to the local time of the receiver. Thus, when a node receives the message, it obtains two time values corresponding to the same time instant: `rootEventTime`, in the local time of the root, and `eventTime`, in the local time of the node. A series of time value pairs are used to estimate the skew of the local clock of the receiver to that of the root. Consequently, the local time of the root becomes the global time in the network. Sequence numbers are necessary because the network-wide broadcast from the root can trigger the same mote multiple times.

Linear regression is commonly used to estimate clock skews (Section II.1.3): a synchronization point is defined as `(local-global-time-offset, local-time)` pair and a fixed number of synchronization points are locally stored in a table at each node. A linear regression line is fit to the data and the slope and intercept of the line are used to estimate the global time in the future. Since clock drifts may change over time, the root needs to re-broadcast synchronization messages periodically. Our ETA-based implementation of RATS

Program 6 Pseudo code of the RATS implementation.

```
struct TimeStampedRATSMessage {
    int    seqNumber;
    int    rootEventTime;
    int    eventTime;
}

OnRootSend() {
    lastSeqNumber++;
    TimeStampedRATSMessage msg;
    msg.seqNumber = lastSeqNumber;
    msg.eventTime = GetLocalTime();
    msg.rootEventTime = msg.eventTime;
    MessageSend(msg);   //broadcast over radio using ETA
}

event Radio.OnReceive(TimeStampedRATSMessage msg) {
    if (msg.seqNumber != lastSeqNumber){
        lastSeqNumber = msg.seqNumber;
        int clockOffset = msg.eventTime - msg.rootEventTime;
        LinRegression (clockOffset, msg.eventTime);
        MessageSend(msg);   //broadcast over radio using ETA
    }
}
```

can be found in Program 6.

The rate with which the root node initiates synchronization rounds influences accuracy, convergence, and overhead of RATS. High duty cycle allows for rapid convergence, but strains the system resources. The best results can be achieved when the rate of time synchronization messages changes dynamically: a high duty cycle phase right after the startup allows for rapid convergence, whereas low duty cycle later on allows for good time synchronization accuracy and resource utilization at the same time. Therefore, our implementation of RATS allows to set the time synchronization duty cycle externally.

2.9.3 Distributed proactive time synchronization (FTSP)

The dynamic nature of ad-hoc radio links and the failures of individual sensors is a well known problem in the WSN domain. It is imperative that WSN algorithms be able to self-configure to allow for automated ad-hoc deployment and to reconfigure to tolerate sensor failures and network topology changes. RATS protocol fails to provide reliable synchronization in certain WSN deployments (e.g., an ad-hoc deployment in a remote area), since the root node is a single point of failure.

We propose the ***Flooding Time-synchronization Protocol***, a proactive time synchronization service that implements the ***Virtual Global Time Service***. FTSP achieves robustness against node and link failures by utilizing periodic flooding of synchronization messages and an implicit dynamic topology update. A node becomes synchronized by estimating the skew of its local clock to the global time. The global time is driven by the local clock of the *root* which is one of the sensor nodes, elected and possibly dynamically reelected by the network. The skews of the local clocks are estimated by measuring a number of *reference points* defined the same way as in RATS: `(local-global-time-offset, local-time)` pairs. Generating reference points in FTSP is a little bit more complex than in RATS: the sender of the time synchronization message records the current local time t_L and calculates the corresponding global time t_G (the sender is synchronized). The sender then sets the value of `eventTime` field of the time synchronization message to $t_L - t_G$ and schedules its transmission using ETA. At transmission, `eventTime` is updated in ETA by subtracting it from the local time of message transmission tm_L and thus the `eventTime` is converted to the global time of the message transmission[2] $tm_G = t_G + (tm_L - t_L)$. At reception, ETA updates the `eventTime` by subtracting it from the local time of message arrival. Consequently, the receiver obtains the reference point `(local-global-time-offset,local-time)` from `msg.eventTime` and `msg.arrivalTime`. Nodes in the broadcast radius of the root collect reference points directly from the root. Other nodes gather reference points indirectly through the synchronized nodes located closer to the root. When a node collects enough consistent reference points, it estimates the skew of its own local clock using linear regression (similarly to the RATS implementation) and becomes synchronized. Consequently, the new synchronized node broadcasts synchronization messages to other nodes in the network.

Program 7 FTSP: time synchronization message format.

```
struct TimeStampedFTSPMessage {
    int    rootID;
    int    seqNum;
    int    eventTime;     //contains global time of transmission
}
```

Synchronization message format: The message contains the `eventTime` mandated by ETA and two extra fields (see Program 7): the `rootID` and the `seqNum`. The `eventTime`

[2]This assumes that $t_{mG} - t_G \approx t_{mL} - t_L$ which only works if $tm_L - t_L$ is small, otherwise the difference of the local clock rate from the universal rate can introduce large errors.

contains the global time estimate of the message transmission according to the sender. The `rootID` field contains the ID of the root, as known by the sender. `seqNum` is a sequence number set and incremented by the root when a new synchronization round is initiated. Other synchronized nodes set the `seqNum` to the most recent (i.e. the largest) `seqNum` value they received so far.

Program 8 FTSP: dissemination of time synchronization messages.

```
1   event Radio.receive(TimeStampedFTSPMessage msg)
2   {
3       if( msg.rootID<myRootID )
4           myRootID = msg.rootID;
5       else if( msg.rootID>myRootID || msg.seqNum<=highestSeqNum )
6           return;
7
8       highestSeqNum = msg.seqNum;
9       if( myRootID<myID )
10          heartBeats = 0;
11
12      if( numEntries>=ENTRIES_LIMIT && getError(msg)>ERROR_LIMIT )
13          clearRegressionTable(); //node or clock failure
14      else
15          LinRegression(msg.eventTime, msg.receptionTime);
16  }
```

Managing redundant information: Since all synchronized nodes periodically transmit time synchronization messages, in a dense network a receiver may receive several messages from different nodes. Due to limited resources, an appropriate subset of the messages must be selected to create reference points. For example, an eight-element regression table stores the selected reference points in the Mica2 implementation. To achieve more accurate clock skew estimation using the limited amount of data, it is more beneficial to store reference points that are distributed over a longer period of time. Therefore, the FTSP protocol selects only the first message that arrives in each time synchronization round. The corresponding code is presented at lines 5-8 in Program 8 and lines 10 and 16 in Program 9.

The root election problem: To perform global synchronization, one and only one root is needed in the network. Since nodes may fail or the network can get disconnected, no single dedicated node can play the role of the root. Thus, a robust election process is needed to provide a root after startup, and also in case of root failure. FTSP utilizes a simple election process based on unique node IDs, as follows: when a node does not receive new time synchronization messages for `ROOT_TIMEOUT` number of message broadcast periods, it declares itself to be the root (line 6 in Program 9). After a `ROOT_TIMEOUT` period, there will be at least one, but possibly multiple roots in the network. Whenever a node receives a message with a `rootID` field smaller than its `myRootID` variable, it updates the variable according to the received `rootID` field. This mechanism ensures that roots with higher IDs give up their status and eventually there will be only one root, the node with the smallest ID, in the whole network. The pseudo-code describing this root election protocol is given at lines 3-4, 9-10 in Program 8 and 3-6 in Program 9. The `heartBeats` variable contains the number of message broadcast periods since the last synchronization point was inserted into the regression table.

Program 9 FTSP: handling new time synchronization messages.

```
1   event Timer.fired()
2   {
3       ++heartBeats;
4
5       if( myRootID!=myID && heartBeats>=ROOT_TIMEOUT )
6           myRootID = myID;
7
8       if( numEntries>=ENTRIES_LIMIT || myRootID==myID ) {
9           msg.rootID = myRootID;
10          msg.seqNum = highestSeqNum;
11          currentTime = GetLocalTime();
12          msg.eventTime = currentTime - local2Global(currentTime);
13          Radio.send(msg);
14
15          if( myRootID==myID )
16              ++highestSeqNum;
17      }
18  }
```

Graceful root change: During the election process special care is taken to avoid inconsistencies and maintain the synchronized state of the network as much as possible. Only the root and synchronized nodes, those that have enough entries (at least `ENTRIES_LIMIT` many) in their linear regression table, transmit time synchronization messages (line 8 in Program 9). If a node receives a new reference point that is in disagreement with the previous global time estimates, it clears its regression table (see lines 12–13 in Program 8). Finally, a newly elected root preserves its estimated clock skew (does not clear its regression table) and broadcasts the global time accordingly. This way the network will not get out of synchronization during the reelection phase. Similar problem arises if a new node is introduced with a lower ID than the current root. In this case, the new root eventually declares itself root, but again care is taken to provide a smooth transition. The new root does not start transmitting its own messages for the `ROOT_TIMEOUT` period, while it collects reference points to estimate its clock skew. Thus, the global time broadcasted by the new root will be synchronized to the previous global time.

2.9.4 Synchronized event (SyncEvent)

SyncEvent is an implementation of the ***Coordinated Action Service*** and allows multiple nodes to agree on a future time for the coordinated action. In SyncEvent, ETA is used to calculate the time remaining until the future event, rather then calculating elapsed time from a past event. SyncEvent implementation works over multiple hops and requires an Alarm service that can signal an event at a future local time.

Program 10 Pseudo code of the SyncEvent implementation.

```
struct TimeStampedSyncEventMessage {
    int    seqNumber;
    int    eventTime;
}

bool Schedule(int time) {
    lastSeqNumber++;
    TimeStampedSyncEventMessage msg;
    msg.seqNumber = lastSeqNumber;
    msg.eventTime = GetLocalTime() + time;
    Alarm.set(msg.eventTime);
    MessageSend(msg);   //broadcast over radio using ETA
}

event Radio.OnReceive(TimeStampedSyncEventMessage msg) {
    if (msg.seqNumber != lastSeqNumber){
        lastSeqNumber = msg.seqNumber;
        Alarm.set(msg.eventTime);
        MessageSend(msg);   //broadcast over radio using ETA
    }
}

event Alarm.fired(){
    signal OnSchedule();
}
```

Program 10 describes the SyncEvent implementation: upon receiving a request to schedule a coordinated action, SyncEvent creates a `TimeStampedSyncEventMessage` and sets `eventTime` with the time of the coordinated event. ETA is used in a similar way as in the RITS implementation, the only difference is that the `eventTime` is a future time, thus the elapsed time corresponds to a negative number. On the receiver node, ETA converts `eventTime` to a future time in the local clock of the receiver and sets the `Alarm` component with this time. This continues until all nodes receive the SyncEvent message. Consequently, the `Alarm` component triggers `OnSchedule()` event on all nodes, when their local time reaches `eventTime`.

2.10 Evaluation of Time synchronization Services

2.10.1 Routing integrated time synchronization

Although RITS promises time synchronization with virtually no communication overhead, uncompensated clock drifts can degrade its accuracy. The following theoretical analysis allows us to design strategies to keep the RITS error small.

Error of ETA: Reactive protocols are susceptible to multiple sources of error, the two most egregious ones being the error caused by different clock rates of the nodes and the error in message timestamping. To express these errors formally, we use the following notation: a node can be a receiver r_i or a sender s_i, $i \in \{1, \ldots, N\}$, equipped with a clock source for measuring local time t_i. We denote a fictious universal time with u and the local time of an event E at the i-th node with y_{Ei}. The offset of the local time from the universal time can change over time because the clock rate of a node can differ from the rate of the universal time (which is 1):

$$t_i = \alpha_i u + \beta_i. \tag{II.1}$$

In the ETA primitive, the receiver r_k synchronizes with the sender s_j by receiving a synchronization message m_i at time u_i:

$$y^s_{ij} = \alpha_j u_i + \beta_j + e^s_{ij} \tag{II.2}$$

$$y^r_{ik} = \alpha_k u_i + \beta_k + e^r_{ik}, \tag{II.3}$$

where y^s_{ij} and y^r_{ik} are the message transmission and reception timestamps, and e^s_{ij} and e^r_{ik} represent timestamping errors of the sender and the receiver, respectively.

We assume that the clock frequencies are stable (i.e., α_i is constant) over the short time interval a packet spends at node i. Also α_i are independent random variables from a symmetric distribution with mean one (that is, the *universal time rate*). We impose no assumptions on the initial clock offsets β_i. Further, both e^r_{ik}, and e^s_{ij} are independent identically distributed random variables with zero mean. Since low-power transceivers have limited communication range, we can further assume that the propagation delay between the nodes is negligible, therefore the universal time of sending and receiving message m_i are the same (i.e. u_i).

The ETA primitive performs inter-node conversion of the event time from the timebase of the sender s_j to that of the receiver r_k as follows

$$y_{Ek} = y_{Ej} + y^r_{ik} - y^s_{ij}, \tag{II.4}$$

where y_{Ej} and y_{Ek} are the event times at the sender and the receiver, respectively.

Error along a routing path: In RITS, ETA is applied iteratively as messages are passed along a routing path on the way to the sink. The first node detects an event at time y_{E1}. Consequently, RITS converts this time to the local times of the second, the third node, etc.:

$$\begin{aligned} y_{E2} &= y_{E1} + y^r_{12} - y^s_{11} \\ y_{E3} &= y^r_{23} - (y^s_{22} - y^r_{12}) - (y^s_{11} - y_{E1}) \\ &\vdots \\ y_{En} &= y_{E(n-1)} + y^r_{(n-1)n} - y^s_{(n-1)(n-1)} = y^r_{(n-1)n} - \sum_{i=1}^{n-1} (y^s_{ii} - y^r_{(i-1)i}). \end{aligned}$$

We denote the timestamping error introduced by the i-th node with e_i and define it as $e_1 = e^s_{11}$, $e_n = e^r_{nn}$, and for $i > 1$, $e_i = e^s_{ii} - e^r_{(i-1)i}$. We further use (II.2) and (II.3) to express y_{En}:

$$y_{En} = y^r_{(n-1)n} - \sum_{i=1}^{n-1} \alpha_i (u_i - u_{i-1}) - \sum_{i=1}^{n-1} e_i. \tag{II.5}$$

For the sake of simplicity, let us assume that the message spends almost the same amount of time at each node along the routing path, $u_i - u_{i-1} \approx \tau$. This way we can separate the skew-dependent and the skew-independent time synchronization errors:

$$y_{En} \approx y^r_{(n-1)n} - \tau(n-1) - \tau \sum_{i=1}^{n-1} (\alpha_i - 1) - \sum_{i=1}^{n-1} e_i, \tag{II.6}$$

where the first term is the time when the message arrives at the last node, the second term is the age of the packet in the universal time, the third and fourth terms are the skew-dependent error and the message timestamping (skew-independent) error, respectively.

RITS and TDOA Measurements: In many WSN applications, sensor fusion is based on time differences of arrival (TDOA) of events. Let us assume that an event E was detected at the same time u_E by two nodes r_1 and r'_1, and the two time tags arrived to the data fusion node along two different paths P and P', such that $P = r_1, \ldots, r_n$ and $P' = r'_1, \ldots, r'_m$. Since the final point of both routes is the data fusion node, we know that $\alpha_n = \alpha'_m = \alpha_{df}$. Without loss of generality we can assume that $n < m$. Also $u_n \approx u'_n$ follows from our assumption of $u_i - u_{i-1} \approx u'_i - u'_{i-1} \approx \tau$. Consequently, using (II.5), (II.3), and $u'_m - u'_n \approx \tau \sum_{i=n}^{m-1} \alpha_1$ we conclude:

$$y'_{Em} - y_{En} \approx \tau \sum_{i=1}^{n-1} (\alpha_i - \alpha'_i) + \tau \sum_{i=n}^{m-1} (\alpha_{df} - \alpha'_i) + \sum_{i=1}^{n} e_i - \sum_{i=1}^{m} e'_i. \tag{II.7}$$

The distributions of clock rates (α_i) and message timestamping errors are assumed to be normal, therefore, the expected values of the first, third and fourth terms in (II.7) are all zero. Interestingly, the expected value of the second term is $\tau(m-n-1)\alpha_{df}$. This means that the clock rate error of the fusion node introduces an error proportional to the difference of the delivery times of the messages at the fusion node. The variance of the two skew-dependent terms sums up to $\tau[(n-1)+(m-1)]V(\alpha)$, meaning that the variance is proportional to the time the messages spend at each node and to the aggregate length of routing paths. The message timestamping error has a variance of $[(n-1)+(m-1)]V(e_i)$, which is proportional to the aggregate length of routing paths.

An important special case is when routing paths overlap. Without loss of generality, we assume that first j elements of the paths are the same:

$$P' = r_1, \ldots, r_j, \ldots, r'_{j+1}, \ldots, r'_m.$$

Then the RITS error is:

$$y'_{Em} - y_{En} \approx \tau \sum_{i=j+1}^{n-1} (\alpha_i - \alpha'_i) + \tau \sum_{i=n}^{m-1} (\alpha_{df} - \alpha'_i) + \sum_{i=1}^{n} e_i - \sum_{i=1}^{m} e'_i. \tag{II.8}$$

Since the skew-dependent errors introduced by the nodes on the overlapping portion of the paths cancel out, the variance of this error decreases to $\tau[(n-1)+(m-1)-j)]V(\alpha)$ which is proportional to the routing time of the packet and to the length of the disjoint portions of the routing paths.

Let us discuss the case of overlapping routing paths in more detail: although a large portion of the time synchronization error cancels out, the actual time of the event reconstructed

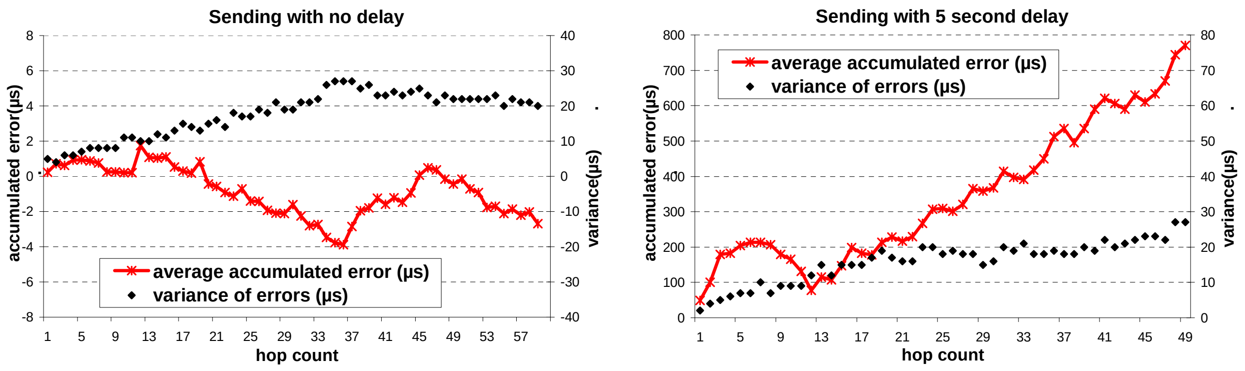

Figure II.4: 50 nodes were arranged on a line: RITS errors at each hop for the no-delay and 5 seconds delay routing strategies are shown. The accumulated errors are averaged over multiple runs and we plot the variance of these errors.

at the sink node may have a large absolute error. Since the variance of the term $\tau \sum(\alpha_i - 1)$ grows with the increased routing time (large τ) (see (II.6)), a large time synchronization error accumulates at the sink. The time synchronization error is small in the TDOA case, because the large skew-dependent error is approximately the same for the two routes and the only major error source is timestamping. Note, that a good short term stability of the clock crystals is a critical requirement for the errors to cancel out.

We verified the stability of the Mica2 clock crystals experimentally: we arranged 50 Mica2 nodes on a line, forming a 50-hop network enforced in software. One of the nodes, the coordinator, broadcasted a RITS packet with its current time and routed the packet to the last node, the sink. The coordinator overheard the packet retransmission at each hop and converted the event time to its local time using ETA. This allowed us to study the accumulated RITS error after each hop. We show the accumulated errors and their variance in Fig. II.4. To make the skew-dependent errors prevail over timestamping errors we introduced a delayed strategy for the RITS packet retransmission: a node waited 5 seconds before retransmitting the packet. The variance of the RITS error is the same in the two experiments, even though the average accumulated error increases in the second case. Therefore, the skew-dependent errors introduce a large, but consistent error, proving that the short term stability of the used clocks is sufficient.

Discussion of theoretical results: We focus our discussion on the applicability of RITS, deriving a set of strategies to keep its time synchronization error small. This analysis is driven by attempting to bound the four error components in (II.8).

(1) First, if it can be guaranteed that the sensor nodes detect an event immediately and the event observation messages are routed to the data fusion node rapidly, RITS will

provide high time synchronization accuracy with no further constraints. This is because the effect of the uncompensated clock drift errors is minimal in this case.

(2) However, if the routing time is long, or the event observations are not routed to the data node immediately after they are detected, highly accurate synchronization cannot be guaranteed. The following strategies can be used to decrease the time synchronization error:

   (a) Data fusion algorithm should be TDOA based. Long routing times can introduce large time synchronization errors which cannot be compensated for at the data fusion node. Even though relating the received event detection times to any global time scale is not possible, the time differences can still be highly accurate. One example of such data fusion algorithm is localization of a source of an acoustic signal from the measured time differences of arrival of the acoustic signal at different sensors [93].

   (b) Routing protocols used with RITS should maximize the overlap in the routing paths. A spanning tree policy with a large tree depth supports the scalability of RITS, while gossip or epidemic protocols can result in dynamic routing paths and therefore should be avoided.

   (c) The data fusion node should be synchronized with a high precision external clock source to limit its clock drift error in case when the time between receiving two observations of the same event is long.

   (d) Finally, localized data fusion should be attempted: detected events should be collocated in time and space, so that the clock drift error of the data fusion node is limited and significant portions of the routing paths overlap.

We support our theoretical results experimentally: we show that RITS can achieve microsecond accuracy, but a small relaxation of the necessary conditions may cause the RITS error to increase.

Experimental evaluation: We tested the performance of RITS in an experimental setup using 45 Mica2 motes arranged in a grid. Each node was enforced to communicate only with its neighbors in software, forming a 10-hop network as shown in Fig. II.5. The test scenario involved multiple observers detecting the same event and reporting the event times to the sink. We simulated the event detection by a radio message that was transmitted by a single node, called the *reference broadcaster*. Since the 10-hop network was enforced in software, all nodes were placed in the radio range of the reference broadcaster and detected

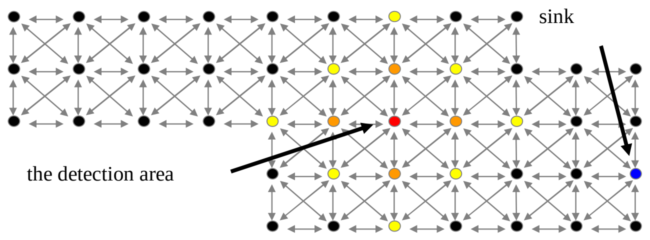

Figure II.5: The layout of the RITS experiment: a node communicates with at most 8 nodes and an event is detected by at most 12 nodes.

the simulated event at the same time instant[3]. To simulate the localized data fusion, the reference broadcaster generated a random event location in the network and only nodes, that were at most 2 hops away from the event location, detected the event. Fig. II.5 shows the sink, the nodes, the radio links and the detection radius of an example event.

For each detected event, RITS packet consisting of the node id (`packet.nodeID`) and the detection time (`packet.eventTime`) was routed to the sink. The sink received detection packets from up to 12 different nodes, each of them containing the event time converted to the local time of the sink. Ideally, these event times should be identical. However, timestamping and clock skew errors cause a small variance of the received times. We evaluated the accuracy of RITS by calculating the maximum and average pairwise difference of all timestamps received by the sink node for a given event, referred to as *maximum* and *average* synchronization errors.

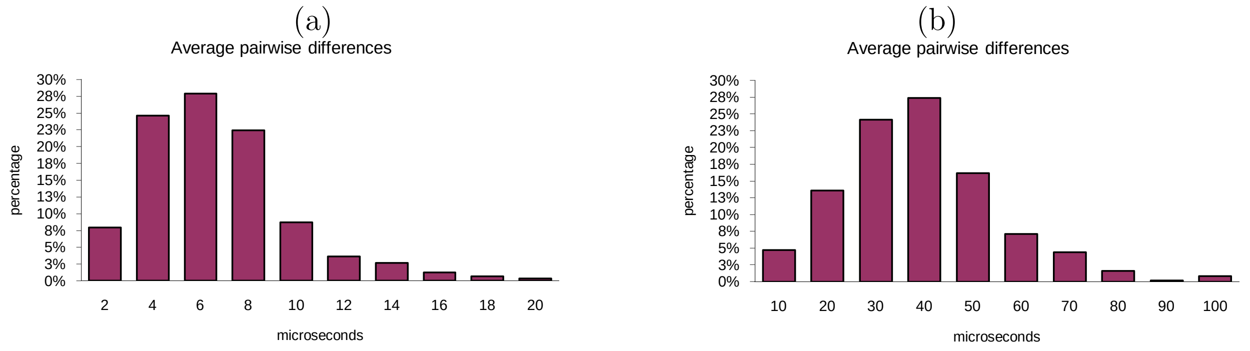

Figure II.6: Histograms of the average pairwise error in the RITS experiment a) with no delay and b) with 5 second per hop delay.

900 simulated events were generated in total and we plot the histogram of the calculated average synchronization error in Fig. II.6a). The average and maximum errors over all rounds were 5.7μs and 80.2μs, respectively.

[3]Ignoring the radio propagation delay.

We have experimentally verified that the error of RITS increases, if the routing time to the sink node increases: we introduced an artificial delay of five seconds between receiving and forwarding the message in the routing layer at each node. Compared to the non-delayed case, the measured average and maximum synchronization errors in an experiment with 700 simulated events grew from $5.7\mu s$ to $34\mu s$ and from $80\mu s$ to $287\mu s$, respectively. The histogram of the average errors can be seen in Fig. II.6b).

To summarize, the performance of RITS is principally affected by two factors. The first error source are the clock skews: we have observed delays of up to 17 seconds between the event happening and the event notification message arriving to the sink in a particular round which is responsible for the large maximum error. The second error source are the timestamping errors: this affects RITS the same way as any other time synchronization protocol that uses radio message timestamping to establish a common time base. The time synchronization accuracy of RITS, as shown by our experiment, is slightly worse than the accuracy of published proactive algorithms [28, 20]. However, we argue that this accuracy is sufficient for most applications and that the energy-accuracy tradeoff of RITS vs. proactive protocols is acceptable, and even desirable, in many cases.

The benefits of RITS over proactive protocols are as follows. First, the clock drifts are not estimated in RITS whereas proactive algorithms require periodic resynchronization to maintain the drifts accurately over time. Second, RITS can be powered down completely and woken up by the event itself, which enables more sophisticated power management techniques than proactive protocols. Proactive techniques prevent the nodes to go to a deep-sleep state, even between synchronization events, since at least the local clock needs to be running. Third, RITS introduces minimal communication overhead; there is no need for time synchronization related messages unlike in proactive algorithms and only a single field needs to be included in the transmitted message.

2.10.2 Centralized proactive time synchronization

The main advantage of centralized synchronization is that a rapid network-wide convergence to a synchronized state is possible through the rapid dissemination of the global time. Distributed protocols, on the other hand, often need to first achieve network-wide agreement on the global time source and only then start the synchronization. Also, since only one node disseminates the global time in the centralized approach, it is easier to control the time synchronization duty cycle allowing for more sophisticated, adaptive power management.

However, unlike fault-tolerant distributed protocols, RATS is vulnerable to the root failure and cannot maintain network-wide synchronization, if the network is partitioned. Often, WSN applications depend on a gateway node which is a reliable node providing a wide-area network access or a PC device connection. In this case, the chance of the root failure can be mitigated.

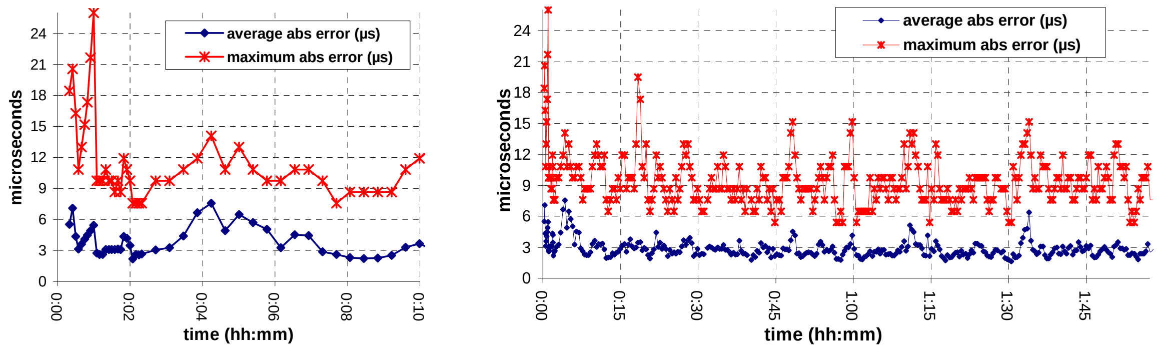

Figure II.7: The RATS experiment showing the maximum and average errors of the reported global times. The figure on the left shows the first 10 minutes and the figure on the right shows the first 2 hours of the experiment. The algorithm converged after 4 seconds and the average error was $2.7\mu s$ in the 11 hop network.

We evaluated the performance of RATS in a test setup similar to the one shown in Fig. II.5. We set up 60 Mica2 motes in a 5x12 grid so that each node had up to 8 neighbors enforced in software. Similarly to the RITS test, a reference broadcast node periodically queried the global times from all nodes and a base station node collected the reported global times. Unlike RITS, here we collected global times from all 60 nodes. The reference broadcaster and the base station were used just for evaluation purposes, they played no role in the synchronization algorithm.

The root transmitted synchronization messages with a period of 2 seconds in the first 5 rounds and then with a period of 30 seconds for the rest of the experiment . The reference broadcaster transmitted the global time queries with a period of 5 seconds during the first 2 minutes and with a period of 23 seconds[4] the rest of the time. Each reference broadcast yielded up to 60 reported global times, one of them being the root's time. For each reference broadcast, we computed the *maximum* and the *average* time synchronization errors as the maximum and the average absolute difference between the root's time and the other times. We ran the experiment for 6 hours and achieved network-wide synchronization in 4 seconds after switching on the root (linear regression requires at least 2 synchronization points). Over all rounds, the maximum and average errors of RATS were $26\mu s$ and $2.7\mu s$, respectively. Fig. II.7 shows the maximum and average errors for each round of the experiment.

At the beginning of the experiment, the errors were generally larger than later on. This

[4]23 and 30 are relative primes, thus the global time was sampled evenly between time synchronization rounds.

is expected as initially only a few datapoints were available for the clock skew estimation. Therefore, the errors of individual measurements had relatively large weight and distorted the clock skew estimates. The data illustrates this point well: when the protocol switched to the 30-second period, the maximum error gradually increased to 26μs and then gradually decreased as the skew estimates improved.

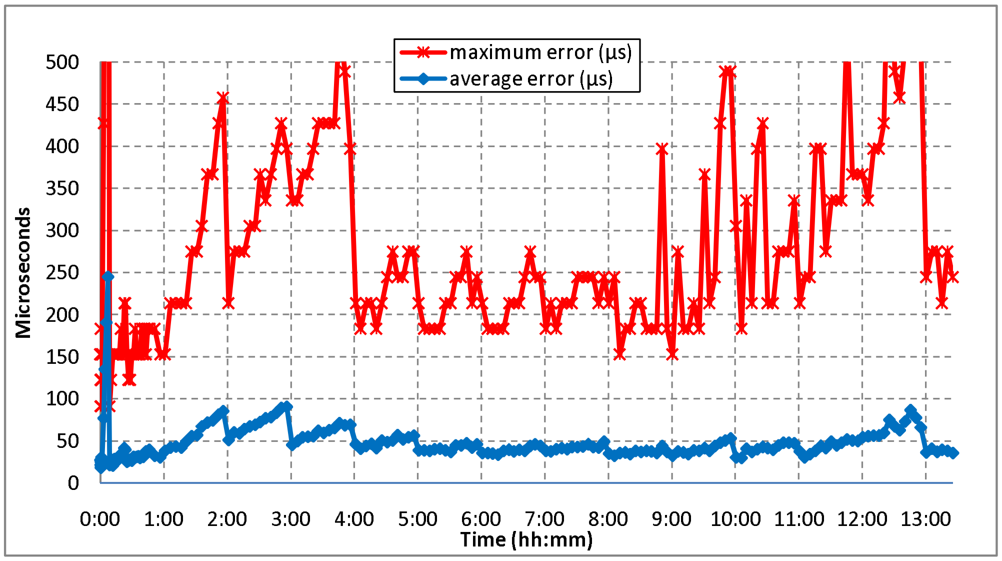

Figure II.8: The energy efficient operation of RATS: synchronization messages were sent once per 1 hour and an internal with the granularity 30.5μs was used. The average and maximum errors over 11 hops were 47μs and 1.3 ms, respectively.

We also tested the potential of RATS to achieve ultra-low duty operation, while keeping a reasonable accuracy. Using the same test setup of 60 Mica2 motes in an 11-hop network, we programmed the root to transmit the time synchronization messages 1) with 10-second period in the first 6 rounds, 2) with 8-minute period in the next 7 rounds, and 3) with 1-hour period for the rest of the experiment. The global time was queried with 6-second, 1.5-minute, and 5-minute periods. We ran the experiment for 13.5 hours and achieved network-wide synchronization 20 seconds after switching on the root. Over all rounds, the maximum and average errors of RATS were 1.3ms and 46.7μs, respectively. Fig. II.8 shows the maximum and average errors for each round of the experiment.

We found that great care is required when changing the duty cycle of the root. If a large change is imposed, the time synchronization accuracy may degrade significantly, up to a point where the network-wide synchronization becomes disrupted. This happens because the clock rates may change over long periods of time (such as 1-hour period) and the clock skew estimates obtained over short time intervals may not be sufficiently accurate. Thus, the error of the linear regression prediction becomes large and the new data is treated as outliers. For example, the average time synchronization error dropped quite rapidly to about $24\mu s$ after 30 seconds. However, 30 seconds later, the root started to broadcast time synchronization messages with 8-minute period and the maximum time synchronization error over this time interval grew to $1.3ms$. The next time synchronization message, coming 8 minutes later, immediately caused the time synchronization error to fall back to $20\mu s$. The effect of changing the 8-minute duty cycle, to the 1-hour duty cycle was less severe, but can be clearly seen in the data at $1:00, 2:00, 3:00$, and $4:00$ times. Despite the $1.3ms$ spike at the beginning, the $47\mu s$ average time synchronization accuracy is an excellent result, considering the $30.5\mu s$ granularity of the clock and the 1-hour duty cycle of synchronization. This accuracy exceeds the time synchronization requirements of the majority of WSN applications.

2.10.3 Distributed proactive time synchronization

Distributed time synchronization algorithms are many times required in ad-hoc deployments of WSN systems. Such algorithms need to be able to self-configure and self-reconfigure to recover from sensor and radio link failures. We subjected the FTSP protocol to a number of challenges, such as switching off the root of the network, removing a substantial part of the nodes from the network, and adding a large number of new nodes to the synchronized network.

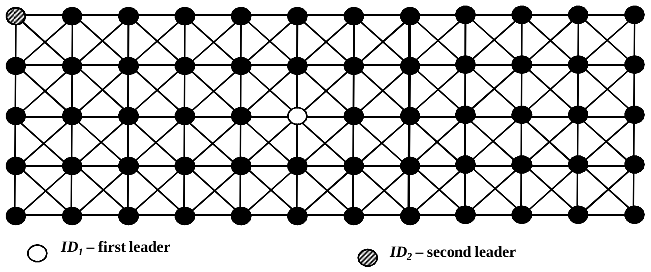

Figure II.9: The layout of the FTSP experiment.

The experiment setup is exactly the same 60-mote setup as in the RATS test. Furthermore, the node with the smallest id (ID1) is located in the middle of the network and the node with the second smallest id (ID2) is at the edge of the network as shown in Fig. II.9. This means that ID1 will become the first root of the network and ID2 will replace ID1 if and when it fails. Note that if ID1 is the root, the location of ID2 represents the worst case scenario in the case the root ID1 dies.

Similarly to RITS and RATS experiments, the accuracy of FTSP was evaluated using periodic reference broadcasts that generated an event, simultaneously observed and timestamped by all 60 nodes. The value of `NUMENTRIES_LIMIT` was 3, and `ROOT_TIMEOUT` was 6, that is, it took a node 6 times 30 seconds, i.e., 3 minutes to declare itself the root if it did not receive new time synchronization messages. The experiment took about 4 hours and consisted of the following steps:

A: at 0:04 all motes were turned on;

B: at 1:00 the root with ID1 was switched off, ID2 becomes the new root, eventually;

C: between 2:00 and 2:15 random nodes were reset one by one with 30 second period;

D: at 2:30 half of the motes were switched off (uniformly distributed);

E: at 3:01 the motes were switched back on (100% new nodes introduced);

F: at 4:02 the experiment ended.

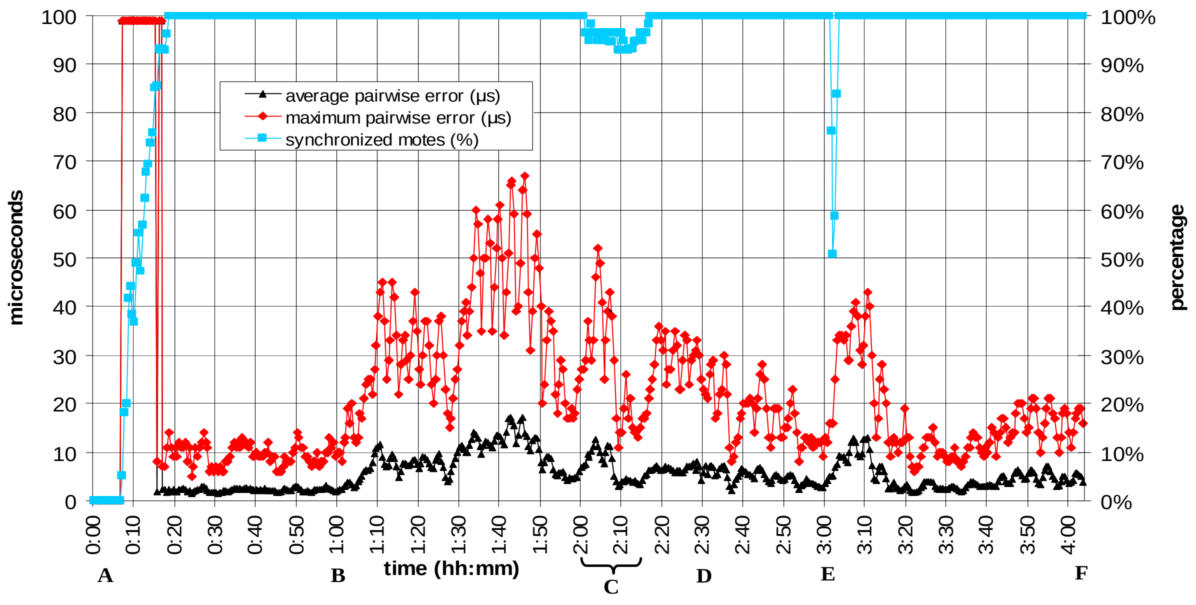

Figure II.10: Experimental evaluation of the FTSP protocol: the percentage of synchronized nodes and the maximum and average time synchronization errors are shown.

Typically less than 5% of the nodes did not reply to the reference broadcaster due to radio collisions. The nodes reported back whether they were synchronized (i.e. had enough values in their regression table) and the global time of the reception of the reference broadcast. For each reference broadcast round, we calculated the percentage of the motes that were synchronized out of those that replied, and calculated the average and maximum time synchronization error as defined in the RITS evaluation. The resulting graphs are shown in Fig. II.10.

The nodes were switched on approximately at the same time (0:04), but during the next 3 minutes (till 0:07) no node was synchronized because none of them declared itself to be the root. Then many nodes timed out and became the roots of the network, which was the reason why the average and maximum synchronization errors soared for 10 minutes until the election process has completed (0:17) and only a single root remained (ID1). The number of synchronized nodes grew steadily during this period and the average and maximum errors became approximately $2\mu s$ and $10\mu s$, respectively. Complete synchronization was achieved in 14 minutes (at 0:18) as indicated by the percentage of synchronized motes reaching 100%.

When the root ID1 was switched off, no impact on the network was immediately observable. What happened is that the global time had not been updated for a certain period of time until each node timed out and declared itself to be the root. The election process again resulted in a single root (ID2) eventually. However, the error stayed low during this time because nodes did not discard their old skew estimates and the new root was broadcasting its estimation of the old global time. This caused deterioration of the maximum and average errors until all nodes calculated more accurate drift estimates based on the messages broadcasted by the new root. From Fig. II.10 one cannot see when the network got synchronized to the second root, but from logged synchronization messages we determined that the election process finished at 1:06 (in 6 minutes).

In the last two parts of the experiment, some of the nodes were removed and new ones were introduced. The impact of these operations on the average and maximum errors was minimal. We can observe that the number of synchronized nodes decreased whenever a new node was switched on because it takes some time for the new node to obtain enough data to get synchronized.

Before we switched off the first root, we had a 60-mote 6-hop network, that is, the maximal minimum distance from the root was 6, for approximately 50 minutes. The average error was below $3\mu s$ translating to a $0.5\mu s$ error per hop. The maximum error was less than $14\mu s$ overall or $2.3\mu s$ per hop.

After the simulated root failure at time B, we had a 59-mote 11-hop network. The average time synchronization error stayed below $17.2\mu s$ for the remainder of the experiment, despite the changes in the topology. If we divide the error by number of hops, we get the average error of $1.6\mu s$ per hop. The maximum error was below $67\mu s$ which was observed only when the root was switched off. Switching off and introducing the new nodes did not introduce a significant time synchronization error.

We have demonstrated that fault tolerance can be achieved without sacrificing the accuracy of time synchronization. In fact, FTSP achieves better accuracy than the centralized RATS protocol (although FTSP synchronized nodes over 6-hop network for the most part, whereas the RATS experiment tested an 11-hop network). However, fault tolerance does not come for free: the convergence time of FTSP is 10 minutes which is significantly larger than the 4 second convergence time of RATS. This is somewhat alleviated by the fact that FTSP allows for graceful change of the root node in case of its failure and minimal effects on time synchronization accuracy were observed when the root was replaced or the topology changed significantly. Another advantage of FTSP is that it distributes the time synchronization related overhead evenly, providing constant radio bandwidth for WSN applications. In contrast, RATS floods the network with time synchronization radio messages, temporarily reducing the available bandwidth of the network.

2.10.4 Synchronized event

The performance of the SyncEvent and RITS algorithms are comparable because both protocols disseminate the time of event rapidly and do not compensate for the clock drifts. The only difference is that SyncEvent disseminates a future time and RITS disseminates a past time. We have evaluated our SyncEvent implementation by connecting two motes to an oscilloscope. When the coordinated action was scheduled, each node sent a signal (i.e., by setting one of its GPIO pins high) to the scope, and the times of arrival differences of the two signals were measured with a very high resolution. The event was scheduled a few hundred milliseconds in the future to keep the clock drift errors low. A few microseconds error was observed over approximately 100 measurement rounds. This shows, that the SyncEvent service is capable of achieving similar performance as the RITS protocol.

We have shown in Section II.10.1 that RITS can introduce large time synchronization error due to the uncompensated clock drifts. Based on the theoretical analysis of RITS, we have introduced a set of strategies to keep the RITS error low which can also be applied to SyncEvent: (1) `TimeStampedSyncEventMessage` messages should be disseminated quickly and the coordinated action should be scheduled as close in the future as possible, (2) the coordinated nodes should be located in the same geographic area, and (3) the global time of the coordinated action may not be accurate. Scheduling the coordinated action in the near future is the most important limitation of SyncEvent. In particular, 10 ppm clock drifts are typical for cheap quartz crystals that are commonly used in WSNs. If the coordinated action is scheduled to happen, for example, 100 seconds in the future, the error can be as high as 1 ms which may be too large for certain WSN applications. Therefore, SyncEvent should be used for scheduling events at most a few seconds in the future, if microsecond accuracy is required.

Chapter III

Space Coordination

3.1 Background

Being able to assign globally consistent locations to the sensor nodes is one of the most fundamental requirements of WSN systems. The sensed data can be tagged with the sensor locations which gives it a context within the physical world. This enables aggregation, correlation, and integration of the data which can significantly improve the accuracy, information density, or dependability of the data measured at a single sensor. Many system level services are designed to use the location information to achieve better power or radio bandwidth efficiency [105]. And finally, localization services are used in many WSN applications, for example, to locate a shooter in a counter-sniper system [93] or a person in an avalanche [75], to perform spatial interpolation of the data collected in the environmental monitoring [71], or to monitor and control traffic and parking [16, 5].

It is usually harder to localize than time synchronize a number of sensors, because there are three dimensions in space as opposed to the one-dimensional time. To estimate the location of a node, the localization service needs to measure low-level spatial relationships between nodes, such as distances or angles between them. These measurements are typically one-dimensional, therefore, at least three measurements to three different *anchor nodes*[1] are required to uniquely localize an unknown node, whereas a single source of global time is sufficient for the majority of time synchronization algorithms. Also we can accurately measure the time change with low-cost quartz crystals, whereas no such measurement method for distances (movement) exists.

3.1.1 Types of localization

One of the unique characteristics of the WSN domain is the large variety of requirements and constraints of applications. It is highly improbable that one algorithm can satisfy the localization needs of all deployments. Consequently, a variety of localization approaches were proposed. These localization approaches can be characterized based on the attainable accuracy, scale, or cost; based on whether they compute physical position (latitude,longitude), or just a symbolic position (kitchen, mailbox,...); whether absolute or relative location is

[1]Anchor node is a node with known location in an external coordinate system.

computed; whether the location is computed externally, or on the devices; and whether the cooperation of the localized object is required [38].

Localization systems most notably differ in the physical phenomena they use to determine the spatial relationships between the nodes. A wide range of these phenomena, such as audible sound [51], ultrasound [85], radio waves [4, 30, 104, 61], infrared light [99], or visible light [94] were studied. The particular choice of the physical phenomenon influences the attainable accuracy, power-efficiency, and the hardware requirements of localization.

Interestingly, the localization problem can be avoided, or mitigated in some WSN systems. These systems can rely on careful surveying of the deployment area to pre-calculate the locations or at least pre-calibrate the system prior to the deployment [35, 99]. Many systems, however, are deployed in an ad-hoc way in inhospitable terrain, for example, dropped from an airplane, where we have no control over the locations of the deployed nodes at all. Similarly, large-scale deployments comprised of hundreds or thousands of sensors prevent precise positioning of individual sensors. Thus, self-configuring and autonomous localization service is required in some scenarios.

Localization for traditional systems: There are several factors that influence the design of WSN algorithms: the small size of the sensors, resource constraints, the use of cheap and disposable hardware components, and energy scarcity. In contrast, almost all traditional localization techniques rely on a relatively expensive infrastructure, require significant computational and power resources and cannot utilize cooperation between localized devices. Consequently, WSN solutions require fresh ideas and novel approaches to overcome these limitations.

The Global Positioning System (GPS) [30] is presently the most popular and widely available technology for localization providing a good solution for many applications. The technique relies on a set of satellites orbiting the earth at known trajectories that are equipped with very accurate atomic clocks synchronized to the global UTC timescale. The satellites transmit radio signals from time-space points that are known accurately. GPS receivers synchronize to the UTC and compute the propagation delay of the signals transmitted by different satellites. Consequently, the distances of the receivers from a number of satellites can be found from where their location can be calculated. The cost of the external infrastructure is justifiable as a one-time investment which becomes insignificant when amortized across millions of different users and scenarios. However, the hardware that is required to compute the location estimate at each GPS receiver is expensive and computation-intensive which makes GPS less desirable for systems where numerous cheap nodes are needed. Moreover, the attainable accuracy of a few meters disqualifies it from high-precision applications. Although differential GPS can achieve better accuracy, it is much more expensive. Finally, GPS requires line-of-sight to at least four satellites, thus it is not available indoors, underwater, or in cluttered urban and forested environments.

Localization in sensor networks: Localization algorithms for WSNs are usually designed as a multi-step process: first, low-level spatial relationships between the nodes, such as distances or angles, are measured. This is called the *ranging* step. Often, hardware manufacturing differences or changing environmental conditions can distort the ranging measurements and need to be compensated for in a *calibration* step. Finally, in the *localization* step, the absolute or relative locations of the nodes are estimated. Due to errors in the ranging data, localization often does not have an exact solution and is formulated as an optimization problem instead. Since the networks can comprise hundreds of nodes, the solution of the optimization problem is beyond the capability of a single node and needs to be solved either centrally on a PC class device, or cooperatively between the nodes in a distributed fashion.

3.1.2 Ranging

Three major factors influence the applicability of a particular ranging algorithm: the accuracy, the maximum measurable range, and the cost (which includes both the power and the hardware cost). The ranging accuracy directly influences the overall localization error, so highly accurate ranging methods are desired. Large maximum range allows for more sparse deployments and cost-efficient solutions. However, both high-level accuracy and long range typically require complex, expensive, and power-consuming ranging hardware, therefore a compromise must be found. The complexity of this task is illustrated by the variety of physical phenomena that were studied and a number of ranging approaches that were proposed by the WSN research community:

- **Range-Free Techniques** do not explicitly measure the ranges between the sensors. Some systems utilize radio connectivity information [36, 10, 80]. Alternatively, passive optical components can be used to localize sensor nodes with no active ranging hardware at all [94]. These techniques provide limited accuracy only, but are simple and efficient.

- **Proximity Based Techniques** use signals with a short range and limited propagation, so that they can be confined to small regions, such as rooms. For example, infrared signals have been used to localize active badges [99]. This approach works because infrared propagation matches topological boundaries of the environment, such as walls. The drawbacks are high installation, configuration, and maintenance cost, because a significant number of sensors are required in each room.

- **Directionality Based Techniques** determine the *bearing* of the sensors with respect to multiple anchor nodes. Both ultrasound and radio signals have been used in this approach. There are two ways to determine the direction of the incoming signal:
  A.) The anchor nodes transmit unidirectional signals. The receiver then determines its bearing relative to the particular anchor node just from the fact that it received its signal. Although the detection hardware on the receivers is simple, relatively high anchor density or the capability of anchors to transmit unidirectional signals in multiple directions are required.
  B.) The anchor nodes transmit omnidirectional signals that spread uniformly in all directions. At the receiver, a relatively complex antenna array or several ultrasound receivers measure the phase difference of arrival or time difference of arrival of the incoming signal and hence, the angle of arrival can be determined [79].

- **Radio Signal Strength (RSS) Techniques** estimate node distances by measuring attenuation of the radio signals transmitted at a known power level. These techniques are hard to implement in the real world: the signal strength of radio signals is subject to unpredictable variations due to channel noise, fading, multipath, and interference coming from the environment. Also, manufacturing differences of radio hardware, antenna shape or length can influence the measured signal strength, thus an extensive calibration is required at deployment [91]. Current state of art work [82] reported 0.9 m to 2.4 m location errors in 81 m^2 area, using 16 motes. The Calamari system [101, 100] localized 49 nodes deployed over 144 m^2 with a median error of 4.1 m.

- **Time of Flight of Sound or Ultrasound Techniques** are also sometimes referred to as time difference of arrival (TDOA) technique. Sound travels in the air much slower than the speed of light. Both radio and acoustic signals are transmitted at the same time and the distance from the transmitter is estimated from the measured time of flight of the acoustic signal. The active Bat System [35] used ultrasound to achieve better than 0.14 m accuracy in 95% cases for room-size ranges. The Cricket System [85] achieved 4x4 feet location granularity in a similar deployment scenario. The ultrasonic version of the Calamari system [101, 100] localized 49 nodes deployed over 144 m^2 with a median error of 0.5 m. The Medusa nodes in the AHLoS system attained 10-15 m range and a few cm accuracy in the lab [91].

- **Time of Flight of Radio Signal Techniques** measure time of flight of radio signals. Since the radio signals travel with the speed of light, these techniques require more sophisticated hardware. Two well known radio-time-of-flight techniques are the GPS and the ultra-wideband (UWB) systems. We have described GPS in Section III.1.1.

UWB is based on very short radio transmissions occupying large bandwidth at high frequencies. UWB systems typically assume that the line of sight signal arrives at the receiver before any environment-reflected signals. This is a reasonable assumption as UWB signals have good object penetration capability due to their frequency diversity. Highly accurate localization is possible because of the high temporal resolution of UWB signals [104]. The problem of both GPS and UWB solutions is that they require accurate time synchronization. An error of 1 μs, which is typical in WSN domain, would translate to 300 m error in ranging. A sensor network solution [61] was proposed that utilizes pairwise round trip time of the radio signals. This technique does not require accurate global time synchronization and was shown to achieve an accuracy of 0.9 m outdoors and 2 m indoors for ranges up to 16 m.

3.1.3 Calibration

Manufacturing differences of hardware components, different environmental characteristics, such as temperature, humidity, or wind can distort the distance or angle measurements. Some of these distortions are systematic and do not change rapidly over time: for example the speed of sound is 331.5 m/s at 0°C and it increases by 0.6 m/s for each 1°C temperature increase. Therefore, the deployed WSN system will consistently underestimate or overestimate the actual measured ranges, if the actual temperature is lower or higher than the expected temperature.

Calibration aims to determine the relation between the measurement standard and the response of the measurement device. A device is typically calibrated in a controlled environment, where a number of output/desired-output data points are recorded. Linear regression, or a simple table lookup is then used to transform the output of the device to the desired standardized output.

Micro-calibration techniques parameterize individual devices based on the model of a particular ranging method. The difficult part is that we do not know how much the transmitter and how much the receiver contributes to the measurement error, which is called the *separation problem*. The SpotON [39] approach first fixes a single transmitter and calibrates all receivers at known distances to that transmitter. In the second iteration, one receiver is used to calibrate the transmission powers of all transmitters.

Micro-calibration is often not manageable in WSN systems, as there are simply too many devices to attend to, and there may be no infrastructural support for the calibration. Therefore, macro-calibration techniques capable of calibrating the whole systems were studied.

Instead of calibrating all pairs of nodes, macro-calibration collects enough ranging measurements to over-constrain the system of equations derived from the parameterized models of interactions between the devices. The least squares method is then used to find the parameters' values, so that the performance of the entire system is optimal [101].

3.1.4 Localization

Localization techniques provide a network-wide, globally consistent map calculated from the calibrated ranging data. Mathematically, the coordinates of the sensors are unknown variables and the measured ranging quantities form a system of equations or inequalities that constrain the unknown locations of the nodes. Localization can therefore be formalized as an optimization problem that minimizes the error of the solution for a given set of constraints. Formally, optimization techniques minimize an objective function, typically defined as the root mean squared error (RMSE)

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x_i})^2,$$

where n is number of measurements, x_i is the measured range, and $\overline{x_i}$ is the corresponding range computed from the localized positions. σ then tells us how well the given solution fits the constraints.

It is a well known problem, that the global optimization of complex problems may fail to converge to an extreme, or may find a local extreme of the objective function. Sometimes an incorrect solution is found due to non-line-of-sight, reflections, and other interferences from the environment. Therefore, localization techniques that are designed as large monolithic algorithms globally optimizing the constraints of the system in a single run often fail. Alternatively, the complex optimization problem can be divided to smaller subproblems which can be solved first and then merged to get the global solution.

Many techniques avoid the local minimum problem by initializing the global optimization problem so that it starts close to the correct solution. If the unknown node is in the range of sufficient number of anchor nodes, its initial location can be found using triangulation, trilateration, or the bounding box method:

- **Triangulation** has been used since the 15-th century to compute distances of ships from the shore. In general, location of a node in n dimensions can be calculated using the law of sines or the law of cosines, by measuring the angles between the node and n reference points and measuring distances between the reference points.

- **Trilateration** has been used less frequently for navigational or geodesic purposes, mainly because it is easier to measure the angles to the landmarks than the distances from the landmarks. In general, the location of a node in n dimensions can be computed by measuring its distance from $n + 1$ reference nodes. Robust trilateration is a special type of trilateration where the locations of the nodes are subject to additional geometric constraints [78]. This technique makes sure by construction, that ambiguous locations are rejected early in the localization process.

- **Bounding Box, or Min-Max** define a bounding box around each reference node and estimate the location of an unknown node at the intersection of the boxes [90, 60]. These techniques are used if trilateration and triangulation are computationally too expensive for the WSN hardware.

However, if the anchor nodes are only accessible through multiple hops, ranges from the anchors need to be determined indirectly. This is achieved by the nodes estimating their spatial relations to each anchor and periodically sharing this information with their neighbors. The following algorithms were suggested based on the types of the ranging measurements that are available:

- **DV-hop** [80] and **Hop-Terrain** [90] utilize the connectivity information. The nodes propagate the minimum hop count from each anchor node. The hop-counts are then converted into the distances by multiplying the hop count with an average hop distance. The average hop distance is inferred by anchors by dividing the known distance between them by the number of hops between them.

- **DV-distance** [80] is similar to DV-hop technique. Instead of the hop count information, it utilizes the distance measurements provided by the RSS ranging method. The disadvantage of this method is that range errors accumulate when the distance information is propagated over multiple hops.

- **DV-position** [80] assumes that both ranging and angle measurements are available. Consequently, the neighbors of the anchor nodes determine and propagate their absolute locations.

The initial locations are typically determined without using all available ranging information. As the next step, refinement techniques are used to further decrease the overall localization error by utilizing all available measurements. Localization algorithms can be divided into two groups, based on whether they solve the global localization problem in a centralized way on a PC class device, or they solve it in a distributed manner on the actual sensor nodes.

- **Centralized Approach**: centralized techniques are relatively simple to implement: ranging measurements are centrally scheduled and then routed back to the central point. Also, it is relatively easy to discover and filter out bad measurements, considering the large amount of ranging data collected in these over-constrained systems. A number of techniques can be used to find the locations: weighted least-squares minimization [58, 25], multi-dimensional scaling [14], and bounding box solution with iteratively shrinking box sizes [90].
- **Distributed Approach**: distributed systems typically sacrifice high accuracy for autonomous operation, power-efficiency, and scalability to a large number of devices. The most popular technique for refining the initial positions is iterative multilateration [78, 92].

Both approaches are susceptible to the potentially high cost of the communication of ranging and location data. In general, if the average hop-count to the centralized computer is larger than the number of iterations required by the optimization, distributed algorithms are more energy-efficient.

3.1.5 Tracking

Tracking algorithms utilize partial location and ranging data to determine the trajectory of mobile targets. Given the locations of a target at past time instances, it is possible to construct a dynamic model of the target movement and reconstruct the target trajectory. Due to the constraints of the WSN domain, both the spatial and the temporal resolution of the target observations are limited: the sampling rate, the length of the observation and the size of the area that can be observed by a single node are limited due to power constraints and the limited range of the sensors. Despite these constraints, WSN systems are often required to provide tracking capability over relatively large areas and long periods of time to support applications such as monitoring, intruder detection, or tactical battlefield surveillance.

The location of an object in the physical world can be identified by sensing different signals, such as acoustic, infrared, or seismic signals, that are generated by the majority of objects. Algorithms for object detection, classification and identification are therefore, important part of tracking systems. In some WSN deployments, we can assume the object's collaboration with the system, simplifying the classification and localization process.

Moving objects generate time-varying signals that can be sampled by the sensors only at a limited rate. Since the measurement of a signal takes non-zero time, the target changes its location during the measurement. Therefore, tracking systems usually divide the global space-time region into *space-time cells* [64] corresponding to the space where the target was located during a given time-period. In general, multiple targets may be present in the area of interest and their trajectories may not be sufficiently separated in time and space. Therefore, tracking systems typically initialize and maintain a track for each target and try to resolve the case when multiple targets occupy the same space-time cell [45].

Practical tracking systems also need to have a built-in mechanism to decide which sensors should be activated to track a particular target. Ideally, we want to minimize the number of active sensors due to power constraints. The problem of deciding which sensor to activate, what should be sensed and where to send the sensed data have been addressed in different ways:

- **Naive, Randomized, and Selective Activation** [81]: in naive activation, all nodes in the network are tracking at all times. In randomized activation, each node is tracking with a probability p. In selective activation, a small subset of nodes track the target, predict the movement of the target, and hand over the tracking to the nodes located at the predicted location.

- **Information Centric Approach** [106]: a single leader sensor node maintains the status of each target and decides which sensors should track the target to obtain the best possible tracking results. The algorithm performs the leader handoff when the target moves.

- **Location Centric Approach** [9]: also called collaborative signal processing, this approach utilizes communication between localized geographic regions, rather than between individual nodes. The choice of the activated nodes becomes simpler to implement which allows to save system resources.

Dynamic models of the movement of tracked objects can improve the accuracy of noisy localization data and help in estimating the expected position of tracked objects. Two major techniques found in the literature are Kalman filters and particle filters:

- **Extended Kalman Filter (EKF)** [47]: the location estimates may by incomplete or noisy, especially, if a high localization rate is required. EKF is used as a post-processing filtering step to obtain a smooth trajectory of the target or to predict the target location. This is achieved by modeling the motion of the target and removing the effects of Gaussian measurement errors. Since the dynamics of the target can vary over time, multimodal version of EKF (multiple EKFs running at the same time) is typically used in real deployments, modeling behaviors such as static target, constant speed, and constant acceleration. It has been observed that although EKF based techniques can provide accurate tracking, they fail to recover from localization errors in certain situations, such as target maneuvering [60].

- **Markov Localization** [27]: it is an extension of EKF that maintains a position probability density for all possible positions of the target. These probabilities are estimated using fine-grained metric discretization of the state space, rather than Gaussian representation of the target's beliefs used in EKF. As a result, multimodal EKF can be represented with Markov localization. The computational complexity of this method is relatively high, especially if high accuracy is required.

- **Particle Filters** [95]: they achieve more accurate results than Markov Localization, but are more computation-intensive. The key idea is to discretize the probability distribution of the object's position rather than maintaining the entire feasible position space. Instead of approximating the probabilities in a closed form using Gaussians, the positions of the object are represented as a collection of particles, each of which has an associated weight. With every new sensor measurement, a new collection of particles is created at the most probable places. Binary sensor networks [3] utilize particle filters where each sensor supplies one bit of information only, such as target is approaching or moving away.

3.2 Related Work

Indoor localization systems:

- **Active Badge** [99]: it was one of the first indoor localization systems. Active badges emit unique infrared signals which are registered by infrared sensors deployed throughout the building. Room level granularity can be achieved, but the system does not work well when sunlight is present and the supporting infrastructure incurs large costs.

- **Active Bats** [35]: it is similar to the Active Badge, but a Bat node transmits an ultrasound signal that is picked up by receivers mounted in the rooms. The location of a BAT node is calculated via multilateration with a few centimeters of accuracy which is much better than that of the Active Badge system. An RF base station coordinates the ultrasound transmissions such that interference from nearby transmitters is avoided. This system relies heavily on a centralized infrastructure and its accuracy is 9 cm in 95 % of the cases.

- **The Cricket system** [85]: it utilizes fixed beacons inside the building transmit ultrasound signals to the localized devices. The actual location is then calculated on the device, making the system more scalable and robust. Cricket can achieve a granularity of 4 x 4 feet.

- **RADAR** [4]: it tracks the users within a building using RF signal strength measurements from fixed base stations. Prior to deployment, a comprehensive set of received signal strength measurements is obtained to build a map of the deployment area. Consequently, the location of a user can be obtained by matching the measured data to the map. This process eliminates multipath and shadowing effects at the cost of considerable preplanning effort. The accuracy of the system is 3 m in 50 % of the cases.

Outdoor localization systems:

- **GPS** [30]: it is a distance based technique that measures time of flight of radio signals. The location of the receiver is computed at the client device using trilateration given the measured distances from at least 4 satellites. The accuracy is between 1 and 3 m 95 % of the time. Differential GPS can achieve 1 cm accuracy, but at a substantially increased cost.

- **CALAMARI** [101, 100]: it combines RSS measurements with acoustic time of flight measurements and performs macro-calibration of the system. Errors were reduced from 75 % of the actual range achieved with no calibration, to 10 % with the calibration. This helped to localize 49 nodes deployed in a 144 m^2 with 0.5 m mean error.

- **AHLoS** [91]: it is a distributed ad-hoc localization system that utilizes anchor nodes as beacons. Iterative multilateration is used to localize unknown nodes using the distance information estimated by RSS or time of arrival techniques. The accuracy of this algorithm was evaluated in simulation, where the distance measurement errors were modeled as Gaussian random variables with 2 cm standard deviation. 10 to 100 node systems localized with 2.8 cm error on average.

3.3 Problem Statement

Many WSN applications require locations of the individual nodes [93, 37, 98]. Despite the considerable research effort invested in this area and the many proposed approaches, robust sensor localization is still an open problem today when applied to real world problems. Techniques based on accurate—typically acoustic—ranging have limited range [58, 101]. They need an actuator/detector pair that adds to the cost and size of the platform. Furthermore, a considerable number of applications require stealthy operation making ultrasound the only acoustic option. However, the range and directionality constraints [85, 102] of ultrasound techniques are even more severe. Methods utilizing the radio usually rely on the received signal strength that is relatively accurate in short ranges with extensive calibration, but imprecise beyond a few meters [102, 39, 103].

The Global Positioning System (GPS) allows to track the location of GPS receivers virtually everywhere on the Earth's surface 24 hours a day. The success of GPS is mostly due to the fact that it works in a global scale and because the service itself is free. With the increased popularity, GPS receivers have become relatively small, cheap, and power conscious. However, GPS does not provide a good solution for certain WSN applications: the accuracy of traditional GPS is a few meters[2], the cheapest chipsets still cost tens of dollars and consume 50-100mW, and finally, GPS does not work well indoors, in cluttered urban environments or under dense foliage.

In summary, the problem with the existing WSN localization techniques is that they have either low cost, adequate accuracy, or acceptable range, but none of the existing techniques satisfies all three criteria at the same time. Since many practical deployments of WSN applications mandate all three criteria to hold, current localization techniques have questionable utility for real world WSN applications. We propose to build a set of localization and tracking algorithms that attain high accuracy and long range simultaneously, while utilizing low-cost, low-power, off-the-shelf WSN hardware.

3.4 Organization

We first describe a novel radio interferometric technique that forms the basis of our ranging, localization, and tracking algorithms in Section III.5. We further improve the interferometric technique to allow for long ranges and better robustness to measurement errors in Sections III.6 and III.7. Finally, we improve the time required to get a position fix to a few seconds, describe a Doppler-effect based algorithm to calculate the velocities of the nodes and implement two real-time location and velocity tracking services in Sections III.8 and III.9.

3.5 Radio Interferometric Measurements

We propose to exploit interfering radio waves emitted from two locations at slightly different frequencies to obtain the necessary ranging information for localization. The composite radio signal has a low beat frequency and its envelope signal can be measured with low precision RF chips using the received signal strength indicator (RSSI) signal. The phase

[2]Differential GPS can get 2 orders of magnitude more precise, but it relies on expensive hardware.

offset of this signal depends on many factors, but the relative phase offset between two receivers depends only on the four distances between the two transmitters and two receivers and on the wavelength of the carrier frequency. By measuring this relative phase offset at different carrier frequencies, one can calculate a linear combination of the distances between the nodes, and ultimately infer their relative position.

The radio RSSI circuitry can be modeled in the following way: the RSSI signal is the power of the incoming signal measured in dBm after it is mixed down to an intermediate frequency f_{IF}. It is then low pass filtered with cutoff frequency f_{cut} ($f_{\mathrm{cut}} \ll f_{\mathrm{IF}}$). Let $r(t)$ denote this filtered signal. We will use capital roman letters to denote nodes. The distance between nodes X and Y will be denoted by d_{XY}. The speed of light is c.

Theorem III.5.1 and its proof can be found in [69]. This theorem provides a formula for the measured relative phase offset.

Theorem III.5.1. *Assume that two nodes A and B transmit pure sine waves at two close frequencies $f_A > f_B$, and two other nodes C and D measure the filtered RSSI signal. If $f_A - f_B < 2$ kHz, and $d_{AC}, d_{AD}, d_{BC}, d_{BD} \leq 1$ km, then the relative phase offset of $r_C(t)$ and $r_D(t)$ is*

$$\vartheta_{ABCD}(f) = 2\pi \frac{d_{AD} - d_{BD} + d_{BC} - d_{AC}}{c/f} \pmod{2\pi}, \tag{III.1}$$

where $f = (f_A + f_B)/2$.

For any four nodes A,B,C and D we define the principal ranging data of our radio interferometric technique, the *q-range*:

$$q_{ABCD} = d_{AD} - d_{BD} + d_{BC} - d_{AC}. \tag{III.2}$$

Equation (III.1) can be rearranged to express the q-range q_{ABCD} using $\lambda = c/f$ and $n \in \mathcal{Z}$:

$$q_{ABCD} = \vartheta_{ABCD}(f)\frac{\lambda}{2\pi} + n\lambda. \tag{III.3}$$

We can use Equation (III.3) to reconstruct the value of q_{ABCD} by analyzing relative phase offsets of the interference signal at different carrier frequencies (different λ).

3.5.1 Sources of ranging error

We briefly discuss the expected sources of error of interferometric measurements in this section. Two nodes transmit unmodulated radio waves at two close frequencies and two receivers measure the absolute phase offset of the received signal $r(t)$. We assume that the absolute phase offset is measured at a fixed time instant that is established between the receivers using some form of time synchronization. The relative phase offset is then calculated by subtracting the absolute phase offsets of the two receivers. The sources of error are the following:

Carrier frequency inaccuracy: It corresponds to the difference between the nominal and the actual carrier frequency of the transmitted signal. According to Theorem III.5.1, the phase measurement error introduced by an average 10 kHz carrier frequency inaccuracy for q-ranges q_{ABCD} less than 100 m is

$$\left|2\pi\frac{q_{ABCD}}{c/f} - 2\pi\frac{q_{ABCD}}{c/(f+10000)}\right| \leq \frac{2\pi\cdot 0.1\text{ km}}{c/10\text{ kHz}} = 0.33\%\cdot 2\pi,$$

independently of the value of f.

Carrier frequency drift and phase noise: Theorem III.5.1 relies on the fact that the frequencies of the emitted signals are stable. Any phase noise or carrier frequency drift will be directly observable in the measured phase offset of the envelope signal. This source of error can be minimized by shortening the length of a single phase measurement, and by minimizing the chance of mechanical, electrical and other kinds of shocks the RF chip is subjected to. In our implementation one measurement lasts for 29 ms so frequency drift is likely negligible. Phase noise, on the other hand, may have a significant effect on phase measurement accuracy.

Multipath effects: The RF signal takes different paths when propagating from a transmitter to a receiver, causing amplitude and phase fluctuations in the received signal. In Theorem III.5.1 we assumed that the radio signal travels from A to C for exactly d_{AC}/c seconds, but this assumption does not hold in the presence of multipath fading. We expect that in many cases higher level algorithms can filter out inconsistent phase offset measurements corrupted by multipath effects.

RSSI measurement delay jitter: The jitter of the delay between the antenna receiving the radio signal and the RF chip delivering the RSSI signal to the signal processing unit would introduce a relative phase offset error at the two receivers. According to our experiments, the jitter is not noticeable.

RSSI Signal-to-Noise Ratio (SNR): SNR is the signal strength relative to noise of the $r_C(t)$ signal. The SNR mainly depends on the physical distance between transmitters and receiver, as the amplitudes of the transmitted signals are exponentially decreasing in space. The SNR value also depends strongly on the hardware implementation of the RSSI detector circuit at the receiver.

Signal processing error: The error introduced by the signal processing algorithm that calculates the phase offset of $r_C(t)$. This is a noisy, logarithmically distorted, low frequency since wave, that can be approximated by a sine wave with additive noise. The frequency and phase estimation of sine waves is a well-studied problem (see e.g. [87, 96]). The theoretical Cramér-Rao bound can be approximated by various signal processing algorithms for a given SNR.

Time synchronization error: The error in the time instances when the receivers measure their absolute phase offset. On representative hardware it is possible to establish a synchronization point with better than 2 μs precision utilizing a single radio message (see Section II.7). Assuming a 2 kHz interference frequency, this translates to $0.4\% \cdot 2\pi$ phase offset error.

3.5.2 The key enabler: the CC1000 radio chip

Mica2 [73] is a hardware platform widely used in the WSN research community. Mica2 motes have the Chipcon CC1000 radio chip which can be configured to transmit in either the 433, 868, or 916 MHz frequency band. Our particular motes were configured to transmit in the 433 MHz band. The following features provided by the radio chip were essential to the implementation of our interferometric algorithm:

(1) the capability to transmit an unmodulated sine wave in a reasonably wide frequency band (between 400 MHz and 460 MHz), as well as the ability to tune the frequency of the transmitter in fine-grain steps (65 Hz),

(2) the short-term stability of the frequency of the unmodulated sine wave (for less than 29 ms time period),

(3) the relatively precise capture of RSSI with a small measurement delay jitter, and

(4) the capability to transmit at different power levels.

The relatively wide frequency band is necessary for calculating the actual q-range from the phase offset differences. The fine-grained frequency tuning is needed to achieve the separation of the two transmitters as required by Theorem III.5.1 in Section III.5. The short-term stability of the carrier frequency and the timing precision of the measured RSSI signal are critical to minimize errors in the relative phase offset. Finally, transmission at different power levels is required since the distance of the two transmitters from a receiver can vary up to the point where the closer transmitter completely overwhelms the signal from the more distant one and no interference signal could be observed.

Even though the radio chip is highly configurable with many favorable properties, it has certain limitations that we had to overcome:

(1) we have observed a large jitter (a few ms) in the time required to calibrate the internal hardware components of the radio before transmitting at any given frequency because of the Voltage Controlled Oscillator (VCO) and Phased Locked Loop (PLL) circuits,

(2) the frequency synthesizer cannot achieve absolute frequencies very accurately, we have seen carrier frequency deviations of up to 4 kHz range that depended on the temperature, voltage levels, and manufacturing differences.

3.5.3 Implementation

We have used the Mica2 mote platform with the TinyOS operating system [42, 84] for our prototype implementation. Our primary objective is to have two nodes transmit sine waves at close frequencies to produce the interference signal and at least two receivers to calculate the phase offset difference of the observed signals. Due to the modulo ambiguity in Theorem III.5.1, it is necessary to measure phase offset differences at multiple frequencies to be able to calculate the q-range. Further, to overcome the limitations of the CC1000 radio chip calibration, we have implemented 1) an external scheduling protocol which drives the interferometric measurement and compensates for the chip's timing fluctuations, and 2) a tuning algorithm that determines the radio parameters for the transmitters so that the resulting beat frequency is in a well defined range (i.e., in a predefined interval few tens of Hz wide).

In particular, our prototype implementation of interferometric measurement consists of the following steps:

(1) selecting a pair of transmitters from a group of participating motes,

(2) fine-grain calibration of the radios of senders to transmit at close frequencies,

(3) time-synchronizing all participating motes,

(4) transmission of a pure sine wave by the two senders at multiple frequencies,

(5) analysis of the RSSI samples at each of the receivers to estimate the frequency and phase offset of the signal at a predefined time instant, and

(6) calculation of the actual q-range from the measured relative phase offsets for each pair of receivers.

We now describe the most important software components of our prototype implementation:

Custom CC1000 radio driver: Our custom driver is designed to co-exist with the standard TinyOS radio driver, so both of them are loaded in the operating system at the same time. A pure sine wave can be transmitted/received at a frequency f the following way: first, the `acquire` command is called to acquire the radio hardware from the standard radio driver, then the `calibrate(t)` command is called to re-calibrate VCO and PLL, so that the radio chip can lock to a new frequency f. Next, the interference signal is generated/received using the `transmit`/`receive` commands, and finally, the control of the radio chip is restored back to the standard driver using the `restore` command.

The self-calibration of the radio chip takes approximately 34 ms for the transceiver to complete, and has to be repeated every time when a new operating frequency is set. The actual transmission time is 29 ms which corresponds to receiving 256 RSSI samples at 9 kHz sampling rate [3]. Therefore, calibration significantly increases the overall measurement time, especially when the measurement is repeated at multiple frequencies. Therefore, for time-critical WSN applications (such as tracking applications), we implemented a more time-efficient version of the radio interferometric measurement:

We found that the calibrated settings for the VCO and PLL do not change significantly for a few hours of continuous operation. Therefore, we self-calibrate the radio chip for all radio channels at start-up, store the calibration values in local memory, and use the recorded values to rapidly set up the radio for consequent radio interferometric measurements at different frequencies. The price we pay is a) larger initialization time of the node (up to two seconds) and b) a dedicated memory buffer at each node to hold calibration values (up to a hundred bytes to cover the whole frequency band). This is, however, well worth the achieved 50 % improvement of the overall measurement time.

Time synchronization: In order to measure the relative phase offset of the RSSI signals between different receivers, the nodes need to measure the absolute phase offsets relative to a common time instant. After collecting the absolute phase offsets of the receivers, the relative phase offsets can be calculated by subtracting them from each other. In this section, we discuss the necessary time coordination strategy. We do not use network-wide time synchronization. Instead, we synchronize the nodes participating in the current ranging round and only for the duration of a single measurement, using our implementation of the *SyncEvent* coordinated action service (Section II.9.4).

Before a node can transmit or receive a radio signal at a particular frequency, it needs to acquire the radio chip from the standard MAC layer and calibrate it to that frequency. Once

[3] We have experimentally determined that 256 samples were required for accurate signal analysis.

the chip is acquired, no further inter-node communication is possible, so the participating nodes have to follow a predefined schedule starting from a precise time instant to stay synchronized. This scheduling is challenging because the time required to perform most operations with the radio, such as `acquire` and `calibrate`, have a significant variance. The crucial operation is the sampling of the RSSI signal that needs to be aligned with a couple of microseconds precision across the receivers.

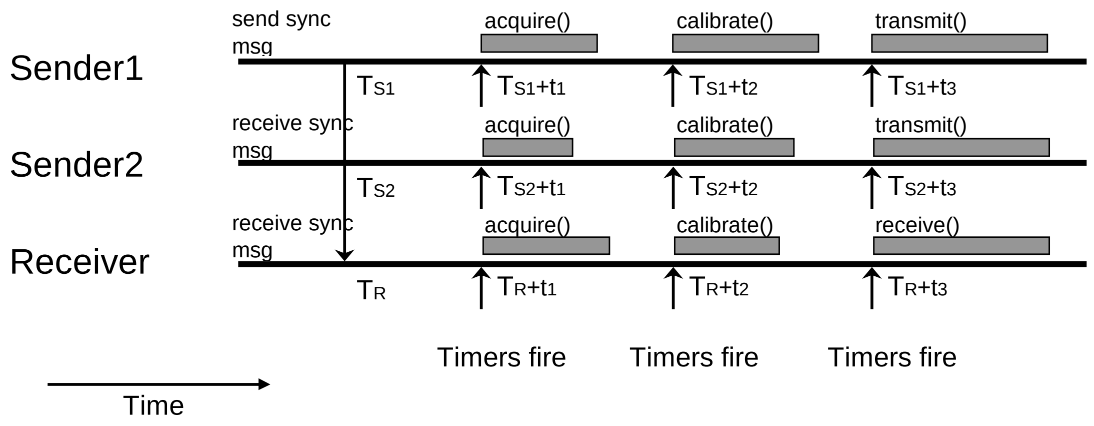

Figure III.1: The synchronization schedule that aligns the start of the transmission and reception at multiple nodes.

The timing uncertainties in the radio chip are mitigated by imposing an external synchronization protocol depicted in Fig. III.1. The first synchronization point ('Timers fire') in this figure is achieved by the SyncEvent component and the firing of Timers at calibration and signal transmission/reception starts at fixed times after the first synchronization point. The combined error of synchronization and clock drift over the measurement period of less than 1 second is still just a few microseconds.

Beat frequency calibration (tuning): As discussed before, the CC1000 radio chip needs to perform internal calibration of the frequency synthesizer PLL to adjust it to the frequency and compensate for supply voltage and temperature variations. Moreover, since the PLL can generate only a limited scale of frequencies, it is advised to recalibrate it for frequencies more than 1 MHz apart [12]. We found it useful to index the available frequency band (400 MHz to 460 MHz) by frequency *channels*, so that transmitting on a different channel mandates re-calibration. We define channel 0 to be at 430.1 MHz and the channel separation to be approximately 0.526 MHz.

A key benefit of CC1000 chip is that the PLL can generate frequencies at a very fine frequency resolution (65 Hz). We index these fine-grained frequencies with frequency *tuning* and require no re-calibration when changing the tuning parameter. The nominal frequency f can be then obtained from the channel and tuning parameters in the following way:

$$f = 430.1 \text{ MHz} + 0.526 \text{ MHz} \cdot channel + 65 \text{ Hz} \cdot tuning.$$

One of the limitations of the chip is that the actual frequency for a specific channel can differ from the nominal value by a few kHz. We need to analyze frequency and phase of the interference signal using the limited sampling rates on the Mica2 platform (9 kHz) over a relatively short time interval. Therefore, the difference between the two transmitted frequencies needs to be in the range of 250 Hz to 450 Hz.

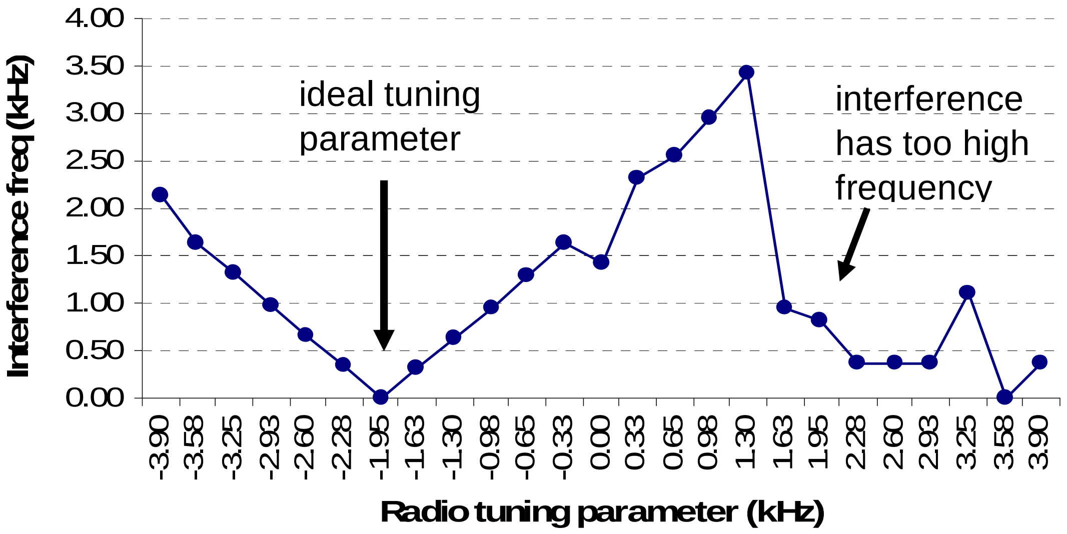

Figure III.2: A receiving node observes the frequency of interference of two transmitters. The first transmitter changes the carrier frequency in 325 Hz steps.

Our beat frequency tuning algorithm calibrates the radio chips of the transmitters, so that the two generated radio signals have almost the same frequency (a few 10s of Hz accuracy is satisfactory). Let f_1, f_2 be the actual frequencies transmitted by the two un-calibrated senders at the same radio channel. Assuming that $|f_1 - f_2| < 4$ kHz we use the fine-grained frequency tuning capability of CC1000 radio chip as follows: one of the transmitters keeps transmitting at frequency f_1, while the other transmitter emits a sine wave at frequencies

$$f_2(i) = f_1 + i \cdot 325 \text{ Hz}, \; i = -15, -14, \ldots, 15.$$

A receiver node analyzes the frequency of the interference signal which is $|f_2(i) - f_1|$ (see Fig. III.2). Using the known step size (325 Hz), the receiver can filter out the noise and faulty frequency measurements and determine the value of i for which the interference frequency is close to 0. The receiver propagates this information back to the transmitters who consequently determine their radio chip parameters, such that the interference frequency is in the required $250 - 450$ Hz range.

The frequency error changes with the absolute value of the transmitted frequency and the difference can be a few hundred Hz for frequencies 50 MHz apart. However, the frequency error is mainly caused by the imprecision of the oscillator that drives the radio chip, thus it is approximately linear in frequency. Therefore, we measure the tuning parameters at two channels relatively far apart and interpolate these values to obtain the radio parameters for the other channels.

Continuous tuning algorithm: The output frequency of the radio transceiver is highly sensitive to conditions such as supply voltage and temperature, therefore, it is not sufficient to execute the tuning algorithm at the application startup only. In fact, the tuning algorithm should be executed every few minutes based on our experience. As the tuning procedure for a given pair of transmitters takes approximately 3 seconds to finish, it conflicts with the real-time requirements of some of the applications.

In certain WSN deployments, it may be possible to completely avoid running the tuning algorithm: the beat frequency measured by the receivers during q-range measurement can be monitored and the tuning values of the transmitters can be updated if the beat frequency drifts out of the operating range. In particular, if the beat frequency is too low (< 300 Hz) or too high (> 400 Hz), the receivers notify the transmitters to update their tuning values accordingly. Consequently, the transmitters are continuously calibrated and there is no need for the tuning procedure (provided the q-range measurement is initiated frequently enough). We have implemented the continuous calibration algorithm for the tracking protocols: tracking requires a new location every few seconds assuring the constant flow of q-range measurements, as well as the constant update of the transmit frequencies.

Frequency and phase estimation: Due to the limited communication bandwidth, the sampled RSSI values need to be processed on the motes. Hardware limitations on the mote make computationally expensive signal processing techniques prohibitive. The ADC sampling rate (9 kHz) and the clock frequency of the 8 bit microcontroller (7.4 MHz), allows roughly 820 CPU cycles per sample for online processing. The lack of floating point hardware support and memory space limitations further restrict the domain of feasible algorithms. Hence, hand optimized fixed point arithmetic is used throughout the frequency and phase computations. The use of standard, but computationally expensive solutions, such as Fourier analysis or autocorrelation, is not feasible.

The signal processing algorithm estimates the frequency and the phase of the RSSI signal in two steps: online and post-processing. The online part is executed upon each A/D converter interrupt for 256 consecutive samples. Because only 820 CPU cycles are available, the online part is relatively simple: first, the raw samples pass through a moving average filter to enhance the SNR and then an adaptive peak detection algorithm is performed. Indexes of the detected peaks are stored in a local buffer.

The post-processing phase determines period lengths from the stored peak indexes. The beat frequency of the RSSI signal is calculated as the reciprocate of the average period length. The phase of the RSSI signal is estimated by the average phase of the filtered peaks. Since small errors in the frequency estimation can result in a significant error in the phase calculation, we compute the phases relative to the center of the sample buffer, thereby reducing the accumulated phase error due to an inaccurate frequency estimate [87]. The algorithm also employs a basic phase unwrapping method to average values near 0 and 2π.

3.5.4 Evaluation

The performance of our frequency and phase detection algorithm is comparable to a high resolution (1 Hz) DFT-based approach. The justification of DFT-based tone parameter estimation and its relationship to the maximum likelihood estimator can be found in [87].

Figure III.3 shows frequency and relative phase offset results of frequency tuning observed on a pair of motes (one of the senders changed its carrier frequency in small increments). The

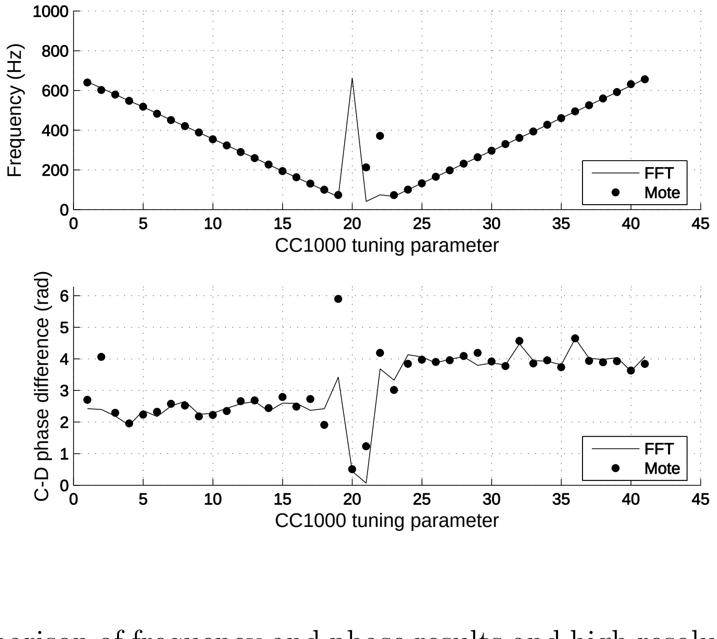

Figure III.3: Comparison of frequency and phase results and high resolution DFT estimation at different interference frequencies.

number of RSSI samples (measurement interval) limits both approaches at low frequencies. In the "normal operating range," however, the mote implementation performs very well. On the frequency diagram both methods closely reveal the ideal "v-shaped" curve. Phase difference measurements have significantly more noise (the ideal response would be a constant value flipped at 0 Hz).

The error of the relative phase offset measurement is illustrated in Fig. III.4. In the experimental setup we fixed a pair of motes as transmitters and performed frequency tuning at 12 additional motes in 30 measurement rounds. Since the positions of the motes were not changing, the error of the relative phase measurement can be found as the average deviation of the measured values over the 30 rounds. We calculated the average deviations for all $\binom{12}{2}$ relative phase offset measurements and calculated the average of these deviations at each beat frequency. This helps us to define the ideal interval for the beat frequency range. The figure also illustrates that by using the signal amplitude value as a quality indicator of the measurements, the average error can be drastically reduced.

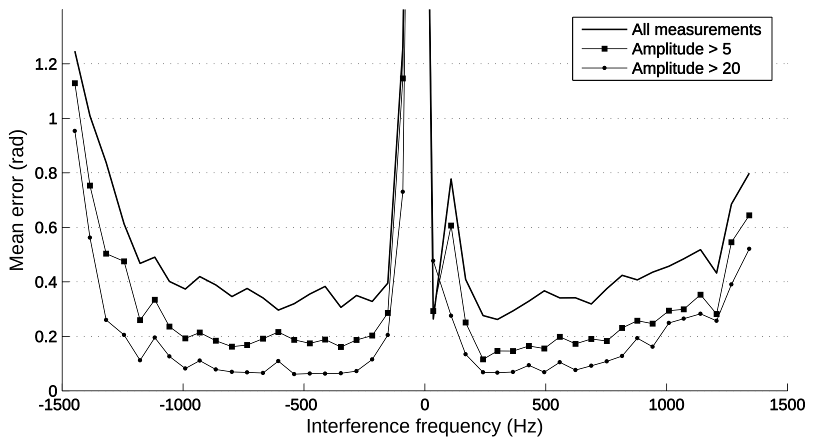

Figure III.4: Mean deviation of phase measurements using different filtering thresholds.

3.6 Radio Interferometric Ranging

Denote the relative phase offset of the receivers X and Y relative to the wavelength of the carrier frequency λ as $\gamma_{XY} = \vartheta_{XY}\frac{\lambda}{2\pi}$, where ϑ_{XY} is the phase offset of X and Y. Equation (III.3) can then be restated

$$q_{ABCD} = \gamma_{CD} + n\lambda,$$

where $n \in \mathbb{Z}$, and both γ_{CD} and λ are known. Clearly, infinitely many q_{ABCD} values solve this equation (n is unknown). We can decrease the size of the q_{ABCD} solution space by measuring the phase offset γ_{CD} at m different carrier frequencies λ_i, $i = 1 \ldots m$. The resulting system of m equations has $m+1$ unknowns, $q_{ABCD} \in [-2d_{\max}, 2d_{\max}]$, where $d_{\max}$ is the maximum distance between any pair of nodes, and $n_1, \ldots, n_m \in \mathbb{Z}$:

$$q_{ABCD} = \gamma_i + n_i\lambda_i, \; i = 1 \ldots m \tag{III.4}$$

Note that this system may still have multiple solutions.

Before proceeding, we give a constraint on the n_i for later use. The phase offsets are less than 2π, so $|\gamma_i| < \lambda_i$. From $n_i = (q_{ABCD} - \gamma_i)/\lambda_i$ and $-2d_{\max} \leq q_{ABCD} \leq 2d_{\max}$, we find $|n_i| < 2d_{\max}/\lambda_i + 1$.

The problem is further complicated by measurement errors. The difference between the nominal and actual radio frequencies for a 50 ppm crystal causes an error in the wavelength of at most $5 \cdot 10^{-5}\ \lambda$, which can be disregarded. The average error of the relative phase measurement on the Mica2 hardware, however, is 0.2 rad or 0.03 λ (see Fig. III.4) and according to our experiments 90-th percentile is more than twice the average error. We denote the maximum absolute phase offset error with $\varepsilon_{\max}$ and rewrite equations (III.4) into the following (implicit) inequalities, $i = 1 \ldots m$:

$$q_{ABCD} \in [\gamma_i + n_i\lambda_i - \varepsilon_{\max}, \gamma_i + n_i\lambda_i + \varepsilon_{\max}]. \tag{III.5}$$

This system has a solution only if the intersection of these m intervals is non-empty. However, it is possible that different assignments to the unknowns n_i solve this system, in which case the system is ambiguous. Note, that the γ_i quantities cannot be controlled. Therefore, if we want to avoid the ambiguity problem, we need to choose values for the λ_i such that ambiguity is excluded. We now derive a necessary condition for ambiguity; by contraposition, its negation is a sufficient condition for avoiding ambiguity.

Assume that also for some different vector of integers $n'_i, i = 1 \ldots m$, the intersection of the intervals $[\gamma_i + n'_i\lambda_i - \varepsilon_{\max}, \gamma_i + n'_i\lambda_i + \varepsilon_{\max}]$ is non-empty, and let q'_{ABCD} be a point in the resulting interval. Putting $p_i = n_i - n'_i$, we have then, $i = 1 \ldots m$:

$$q'_{ABCD} - q_{ABCD} \in [p_i\lambda_i - 2\varepsilon_{\max}, p_i\lambda_i + 2\varepsilon_{\max}].$$

Since the intersection of these m intervals is non-empty, so is the intersection of any pair. This means that, for all pairs i, j in the range $1 \dots m$:

$$|p_i\lambda_i - p_j\lambda_j| \leq 4\varepsilon_{\max}. \tag{III.6}$$

A further constraint on the p_i values, which are integers, is found from the constraint given earlier on n_i: $|p_i| = |n_i - n'_i| < 4d_{\max}/\lambda_i + 2$. The distinctness of the vectors n_i and n'_i, finally, requires at least one p_i to be non-zero.

If, conversely, we can find λ_i such that system (III.6) has no solution in integers p_i—subject to the further constraints—then system (III.4) is guaranteed to be unambiguous for all possible outcomes for the γ_i.

We put this in context by providing concrete characteristics of our radio driver used for the Chipcon CC1000 chip: the frequency range is 400–460 MHz, the minimum separation f_{sep} between the possible frequencies is 0.526 MHz, and $\varepsilon_{\max}$ is 0.075 m.

With these parameters, it is actually impossible to find a set of frequencies for which (III.6) is unsolvable. To start, there are solutions for very small p_i. In particular, taking $p_i = 1$ for all i, insolvability requires that $\varepsilon_{\max} < \frac{1}{4}|\lambda_i - \lambda_j|$ for some pair i, j. But the range of wavelengths is 0.65–0.75 m, requiring then that $\varepsilon_{\max} < 0.025$ m, way below the actually obtainable precision. In practice the situation is not that dire; for this to result in an actual ambiguity, all phase-measurements errors have to "conspire", with those for the smaller wavelengths being high (positive), and those for the larger wavelengths low (negative). For a set of, say, seven wavelengths, this is rather unlikely, although not impossible. Should an ambiguity of this type occur, then at least the error is not extremely large.

Potentially much more pernicious are errors with large values of p_i. To express them, we rewrite system (III.6) into $|p_i(\lambda_i - \lambda_j) - (p_j - p_i)\lambda_j| \leq 4\varepsilon_{\max}$ and then into

$$|p_i - (p_j - p_i)\lambda_j/(\lambda_i - \lambda_j)| \leq 4\varepsilon_{\max}/(\lambda_i - \lambda_j). \tag{III.7}$$

Putting $d = p_j - p_i$, we have then:

$$p_i \in [\frac{d\lambda_j}{\lambda_i - \lambda_j} - \frac{4\varepsilon_{\max}}{\lambda_i - \lambda_j}, \frac{d\lambda_j}{\lambda_i - \lambda_j} + \frac{4\varepsilon_{\max}}{\lambda_i - \lambda_j}].$$

It is easy to see that all large p_i correspond to the integer multiples of $\frac{\lambda_j}{\lambda_i - \lambda_j}$, which means errors of $\frac{\lambda_i\lambda_j}{\lambda_i - \lambda_j}$ in q_{ABCD} solution space. Note, that $\lambda_i\lambda_j/(\lambda_i - \lambda_j)$ corresponds to the wavelength c/f_{sep}, where $f_{\text{sep}} = |f_i - f_j|$ is the frequency separation of f_i, f_j. Fortunately, it is not hard to find relatively small "perfect" sets of frequencies, i.e., ones for which such large errors are excluded. In particular, we use 0.526 MHz frequency separation which has the wavelength of about 570 m, allowing q-ranges up to 285 m.

3.6.1 Multipath effects

As we experimented with extending the range of interferometric measurements, the results quickly deteriorated. Once the cell size reached 10 m in the grid setup, the ranging error distribution got significantly worse. However, if we elevated the motes off the ground, the results improved markedly again. The nodes needed to be less than 10 m apart if they were on the ground, but could be more than 20 m apart if they were 4 feet high on tripods.

Figure III.5 shows representative phase offset measurements on 120 channels in the 400–460 MHz frequency range with the nodes on the ground (a) and 1.3 m elevated (b). Notice that the variance of the measured phase offsets is significantly smaller in the elevated scenario, while there are severe fluctuations in case of ground deployment. When we repeated the same experiment in a rural area far from buildings and trees, the results were close to the ideal case irrespective of the deployment height. The last observation suggests that multipath propagation is at play here, but why does elevating the motes apparently fix the problem?

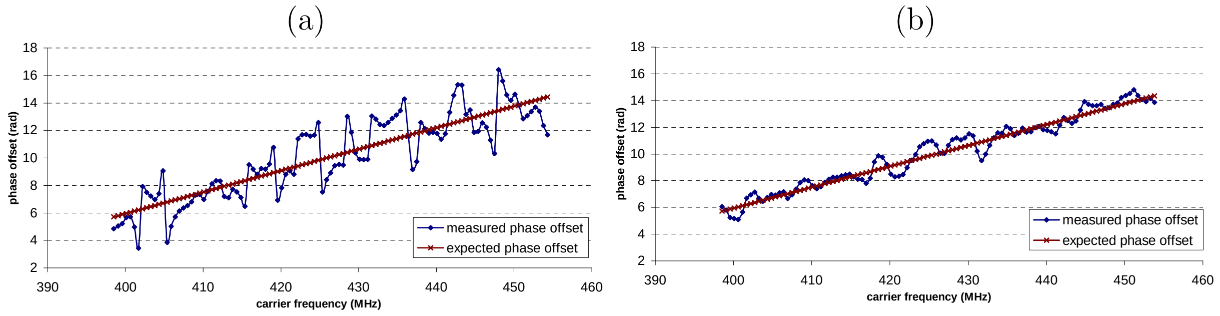

Figure III.5: Phase offset measurements on 120 channels with vertical monopole antennas directly on the ground (a) and 1.3 m elevated (b).

Until now, we have considered the radio nodes as if they were operating in free space. However, the ground around and under the antenna and other nearby objects such as trees or buildings can have significant impact on the shape and strength of the radiated pattern. When the radio wave strikes a surface, it is reflected with an angle that is equal to the angle of incidence. For small angles of incidence, the reflected signal has smaller, but still significant amplitude and the reflected phase is $-\pi$. The distance difference between the line-of-sight (LOS) and the ground reflected signals is small at small angles. Therefore, the phase shift between them remains close to $-\pi$ and hence, the composite signal is attenuated.

We observed that the amplitude of the composite signal grows tenfold when we elevate

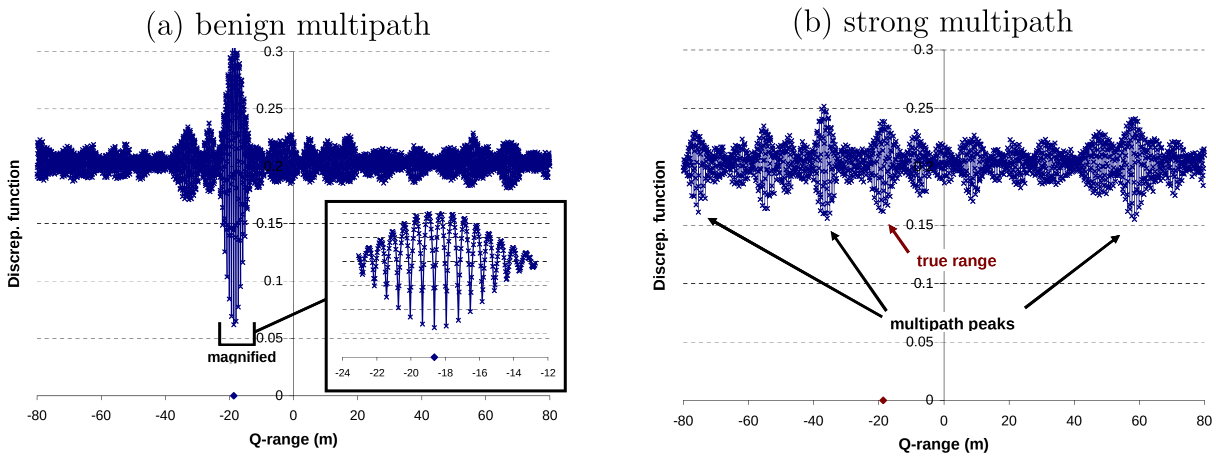

Figure III.6: a) The global minimum of the discrepancy function f is at the true range, but f is close to the global minimum at multiples of $\overline{\lambda} = 70$ cm around the true range. b) Multipath introduces additional errors, resulting in multiple local minima with almost the same value. Global minimum may not be at the true range.

the nodes from ground level to 1 m (at 30 m distance). This attenuation is not a problem in and of itself, as long as we can still measure the phase of the signal accurately. It definitely decreases the effective range of the method, but it does not by itself impact the accuracy noticeably. However, in a moderate multipath environment, such as the campus area, reflections from buildings and other surfaces distort the results. As these reflections travel longer distances, they are markedly attenuated. As long as the direct LOS signal is strong, the additional phase shift these components cause is small. However, when the ground reflection attenuates the LOS signal, the phase shift caused by additional multipath components is large enough to distort the results considerably, as shown in Fig. III.5.

3.6.2 Coping with the ranging error

Intuitively, solving the ranging problem can be thought of as fitting a straight line to the measured data. As shown in Fig. III.5, the ideal phase offset is linear as a function of the frequency if we allow for wraparound at 2π. If we have data distorted by multipath and other errors, we can still fit a line relatively accurately, provided we have enough good data points. Therefore, a trivial enhancement is to make measurements at as many frequency channels as possible. However, this also increases the required time of the actual ranging, and a balance must be struck.

We now show how to estimate the q-range, even in the face of q-range ambiguity and moderate multipath effects. To get the q-range q_{ABCD} we have to solve the inequalities (III.5). Given a possible q-range r, for each inequality (i) we can find the value of n_i that brings $\gamma_i + n_i\lambda_i$ the closest to r, namely $n_i = \text{round}\left(\frac{r-\gamma_i}{\lambda_i}\right)$. We define a *phase-offset discrepancy* *function* as the average value of the squares of $\gamma_i + n_i\lambda_i - r$ values as a function of r. Ideally, the global minimum of the discrepancy function is 0, attained at the true q-range. However, measurement errors, multipath effects and the ambiguity due to the limited number of channels and the minimum channel separation, distort the results. Frequently, the global minimum of the discrepancy function is not at the true solution, but at small integer multiples of the average wavelength $\overline{\lambda}$ from q_{ABCD} (see magnification in Fig. III.6a). Therefore, the distribution of q-ranging errors is not Gaussian, but it exhibits increased error probabilities at small integer multiples of $\overline{\lambda}$ from the mean. The q-ranging error is further influenced by multipath as shown in Fig. III.6b. Even though the discrepancy function has a local minimum at the true q_{ABCD}, the global minimum of f can be located far away.

We aim to eliminate both wavelength and multipath related ranging errors shown in Fig. III.6. Since the global minimum of the discrepancy function might be a false solution and hence, the true solution might be at a local minimum, we define the output of our ranging algorithm as a set of possible q-ranges rather than a single value. The task of choosing the correct q-range from this set is left to the higher level localization or tracking algorithms. Formally, we define the ***q-set*** S_{ABCD} as:

$$S_{ABCD} = \{q \in \mathbb{R}, \forall i : |q \bmod \lambda_{\mathrm{i}} - \gamma_{\mathrm{CD_i}}| < \texttt{windowSize}\}, \tag{III.8}$$

where A, B are transmitters, C, D are receivers, γ_{CD_i} is the relative phase offset of C, D measured at channel with wavelength λ_i, for $i = 1 \dots m$ and `windowSize` specifies how much discrepancy in the phase offset measurements is tolerated. We often refer to the results of our ranging algorithm by “q-range” instead of “q-set” by which we mean the least-error q-range in the corresponding q-set.

If the tracked object is moving during a measurement round, q is not constant in (III.8). If the speed of the tracked object is large, or the measurement takes too long (large m), the observed variance of q may become larger than `windowSize`, resulting in $S_{ABCD} = \emptyset$. One could argue that this could be resolved by setting `windowSize` large enough. However, increasing the window size above $\overline{\lambda}/2$ would mean that every range q would satisfy the constraints of (III.8). Therefore, we needed to constrain the window size and consequently, the maximum speed of the localized object.

3.6.3 Improving ranging accuracy

For a given quad of nodes A, B, C, and D, the q-set S_{ABCD} contains the true q-range q_{ABCD} and a number of incorrect q-ranges, depending on the `windowSize` parameter. We propose an algorithm that can iteratively remove the incorrect q-ranges from the q-set, given that multiple q-sets are measured for a set of nodes whose locations are constrained by their Euclidean space positioning.

Given arbitrary locations of the nodes, we define a *q-range discrepancy* as the average value of the squares of the differences between each measured q-range and the corresponding q-range “on the map”, i.e., the q-range computed from the node locations. If sufficiently many ranging measurements are collected, it is possible to find the node locations by minimizing their aggregate discrepancy of all the q-ranges. After analyzing several experiments, we made the following observations:

First, relatively accurate approximate locations of the nodes can be found even if only a small portion (30%) of q-ranges are accurate;

Second, even in multipath environments, the phase-offset discrepancy function has a sharp local minimum at the true q-range in most of the cases (see Fig. III.6).

These two observations led to the idea of the iterative phase-offset discrepancy correction algorithm:

(1) Let S be a set of all measured q-sets and R be the current search radius, initialized to its maximum value.

(2) Calculate optimal node locations L using the least-error q-ranges from each q-set in S.

(3) Using Equation (III.2), calculate q-range estimates Q from the location estimates L.

(4) Use the set of current q-ranges estimates Q and the radius R as constraints: for each q-set in S, remove all q-ranges that are outside the radius R from the corresponding q-range estimate.

(5) If R is small enough, stop. Otherwise, decrease R and go to step 2.

In other words, the current localization solution is used to remove the incorrect q-ranges from the q-sets in such a way that the large errors are progressively eliminated. Due to this feedback method, the q-range estimates get more accurate at each iteration. In our current prototype, we use fixed decreasing R values, such as $R_1 = 50$ m, $R_2 = 5$ m, $R_3 = 0.5$ m.

3.6.4 Evaluation

We evaluated the accuracy of interferometric ranging in different settings. First, we had a field demonstration at the UC Berkeley Richmond Field Station. We used a 50-node approximate grid setup with a cell size of 9 m in a moderate RF multipath environment. The ground truth was obtained using GPS with an estimated accuracy of 1 m. 68% of the measured q-ranges had less than 1 m error, while 89% was within 2 m. Because of the inaccurate ground truth, these numbers are not revealing[4]. However, the experiment did provide an important datapoint. To get a better assessment of the overall accuracy, we hand-measured a 30-node subset of the network using measuring tape. We estimate the ground truth obtained this way to be about 5 cm accurate. We scheduled 107 transmitter pairs and collected 1392 actual q-ranges from the network, of which 82% had 20 cm error or less, while 95% were better than 1 m accurate.

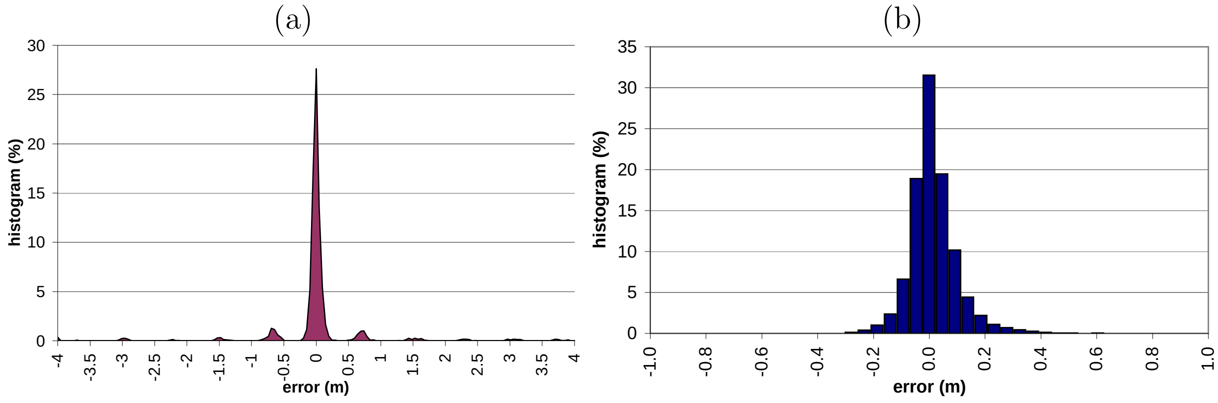

Figure III.7: Error distribution of q-ranges (12000 m^2 experiment) obtained using (a) regular and (b) iteratively corrected ranging.

Finally, we tested interferometric ranging in a rural area where there were no RF multipath effects other than ground reflection, using 16 ExScal motes deployed on the ground in an approximately 12000 m^2 area with an average closest-neighbor distance of 35 m and a maximum node distance of 170 m. We used a laser range finder to obtain the locations of the nodes with an estimated average error of 2 cm. We plot the error distribution for both original and iteratively corrected ranges in Fig. III.7. The original q-range error distribution had only 72% below 30 cm. Within three error-correcting iterations all q-range errors dropped below 1 m, while 98% were under 30 cm.

3.7 Radio Interferometric Localization (RIPS)

Since the interferometric ranging method provides range estimates between a pair of nodes indirectly through a combination of distances among four nodes, none of the existing localization methods is directly applicable. As a first cut at the localization problem, we used a genetic algorithms (GA) based optimization and named our algorithm the *Radio*

[4]Since q-range is a linear combination of 4 distances between nodes, the accuracy of the q-range ground truth can be as bad as 4 times the node location accuracy.

Interferometric Positioning System (RIPS). Although RIPS can provide accurate localization results, it is computationally complex and its convergence time is large. Therefore, the applicability of RIPS to practical WSN deployments is limited and we only use it to evaluate the interferometric ranging method in the context of overall localization accuracy.

Given a set of nodes with unknown locations and a set of measured q_{ABCD} ranges, our goal is to find the relative positions of the nodes that fit the measured q-ranges the best. A standard root mean square error (RMSE) function is defined over the node locations and a GA is used to find the node locations with the smallest error.

A node placement is represented directly by a vector of (x, y, z) coordinates of the nodes. GA starts in with an initial map in which sensors are randomly positioned. The following genetic operators are used to iteratively improve the current map:

(1) Crossing over: position of a node is inherited from one of its parents with even chance.

(2) Mutations (all cases have equal chance)

   (a) Move one node by a Gaussian random number with a small ε variance.

   (b) Move one node to a random position.

   (c) Move all nodes by the same Gaussian random number with a small ε variance.

The value of ε is set to the current value of the RMSE function. It makes it possible for the nodes to do bigger "jumps" if the error is larger. When the error gets small the nodes can fine tune their positions this way.

3.7.1 Evaluation

We ran an experiment using the filtered ranging data shown in Fig. III.7. The RIPS algorithm ran for a few minutes and the results are shown in Fig. III.8. The average and maximum localization errors were 4 cm and 12 cm, respectively, using the three anchor nodes shown by triangles. Selecting the nodes in the four corners as anchors instead resulted in no change in the average error, but a decreased maximum error of 6 cm.

Note that the estimated accuracy of the ground truth and the accuracy of the localization results are comparable. Therefore, these numbers are not exact; they are just indications of the high precision interferometric localization can achieve.

3.8 Radio Interferometric Tracking (mTrack)

In localization, locations of a large number of nodes are determined using a few anchor nodes. In contrast, tracking algorithms can typically make use of significant infrastructure

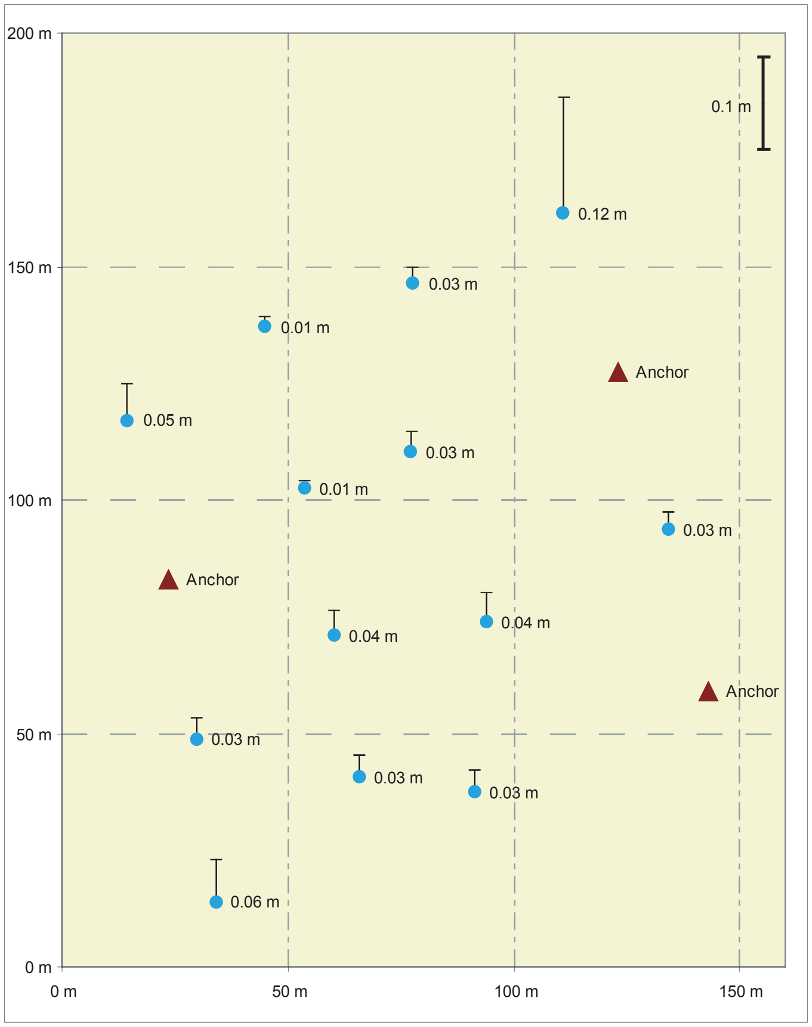

Figure III.8: The setup for the 12000 m^2 experiment showing the anchors (large triangles), motes (small circles) and errors bars.

support, such as GPS satellites, or radar towers. Despite this, tracking is often a more challenging problem than localization because the location of the tracked node can change significantly during the ranging measurement which can result in inconsistent distance measurements.

In our approach, we envision a fixed set of resource constrained wireless devices deployed at known locations forming the infrastructure of the tracking service. Their function is the same as that of the GPS satellites: they act as RF sources for the purpose of locating objects relative to their known positions. The tracked object is assumed to cooperate with the tracking system and its movement is confined to a certain area, so that sufficiently many infrastructure nodes are in its communication radius at all times.

Let us first assume that the tracked node is not moving and address the mobility related problems later. In the stationary case, the location estimation actually becomes simpler than in the localization problem. This is because out of the four nodes participating in the q-range measurement, the tracked node is the only unknown node participating in the q-range measurements. Let us assume, without loss of generality, that node A is the tracked node at an unknown location and nodes B, C, and D are infrastructure nodes at known locations. Since the distances between B, C, and D are known, we can rearrange the q-range equation (III.2) such that the known or measurable quantities are all on the right hand side:

$$q_{ACD} = d_{AD} - d_{AC} = q_{ABCD} - d_{BC} + d_{BD}. \tag{III.9}$$

We call the term on the left hand side the ***t-range*** q_{ACD}. Note that it relates only three nodes, not four. Geometrically, the t-range q_{ACD} represents an arm of a hyperbola (in two dimensions) with foci C, D and with the semi-major axis of the length $q_{ACD}/2$. Similarly, if the unknown tracked node is a receiver C, the following t-range is measured

$$q_{CAB} = d_{BC} - d_{AC} = q_{ABCD} - d_{AD} + d_{BD}. \tag{III.10}$$

Since the t-range can be computed from the measured q-range and the known distances between the infrastructure nodes, each q-range measurement constrains the location of the tracked node to one arm of a hyperbola. Multiple different q-ranges can be measured for a given tracked node, thus the tracked node can be localized at the intersection of multiple hyperbolas.

As we discussed before, to disambiguate the measured q-range for a given pair of transmitters, we need to analyze their interference at multiple carrier frequencies. We call this multi-step process a *measurement round*. It is important to note that the number of relevant q-ranges acquired during a measurement round depends on wether the tracked node is a transmitter or a receiver.

Target as a transmitter: In a measurement round where the tracked node and a designated infrastructure node are the two transmitters and the rest of the $n-2$ infrastructure nodes are receivers, there are $\binom{n-2}{2}$ receiver pairs yielding $\binom{n-2}{2}$ q-ranges in which the tracked node is involved. Assuming no two receivers have the same location, $\binom{n-2}{2}$ unique t-ranges can be calculated, though, only $n-3$ of these t-ranges are linearly independent. This is because each receiver participates in $n-3$ q-ranges and all other q-ranges can be computed as a linear combination of the $n-3$ fixed q-ranges. The large number of q-ranges that can be collected allow us to find the location of the tracked node with high precision at the intersection of the corresponding hyperbolas. The disadvantage of the target-as-transmitter approach is its poor scalability in tracking multiple nodes. To avoid concurrent transmissions by different tracked nodes, exclusive access to the radio channels has to be guaranteed. Therefore, the latency of this method increases proportionally to the number of tracked nodes.

Target as a receiver: If the tracked node is a receiver and two designated infrastructure nodes are transmitting, there are only $n-3$ receiver pairs involving the tracked node, and thus only $n-3$ q-ranges are relevant. Moreover, when we transform q-ranges to t-ranges, all t-ranges will involve the same three nodes: the two transmitters and the tracked node. That is, all t-ranges are the same (not considering the measurement error), defining the same hyperbola. As a result, one measurement round does not yield enough information to calculate the position of the tracked node in the target-as-receiver case. However, this approach allows to track theoretically arbitrary number of nodes, since they do not require to transmit any information on the radio channel as long as each node computes its own location.

3.8.1 Localization of the tracked node

The GA based localization described in Section III.7 is impractical due to its computational complexity and large time convergence. We propose an alternative solution that utilizes the notion of t-ranges, or equivalently hyperbolas, and finds the location of the tracked node at the intersection point of hyperbolas.

We first focus on the simplest case which is finding the intersection points of two hyperbolas. This can be trivially extended to find the intersection of multiple hyperbolas: find the set of intersection points I of all pairs of hyperbolas. The location of the tracked node can be estimated by the center of gravity of I.

In general, finding the intersection of two hyperbolas analytically is hard. We solve a special case instead, which imposes minimal constraints on the topology of the deployed infrastructure nodes: we assume that the two hyperbolas share a focus. This is a reasonable assumption: if the tracked node is a transmitter, each infrastructure receiver participates in $n-3$ q-ranges and thus, it is located at a common focus of the corresponding $n-3$ hyperbolas. If the tracked node is a receiver, multiple measurement rounds are necessary. We can select the transmitter pair in the first measurement round to be nodes A and B and in the second round to be nodes A and C, assuming A,B, and C are cooperating infrastructure nodes. Node A is then a common focus of the two hyperbolas. Since node A always participates in the transmission, it can coordinate the measurement process and we call it the *master* node. The co-transmitting nodes B and C are *assistant* nodes and the tracked node is denoted by X. The notation in the target-as-transmitter case is analogous.

Consider Figure III.9: hyperbola h_{AB} is defined by its foci A, B and the distance R_{AB}

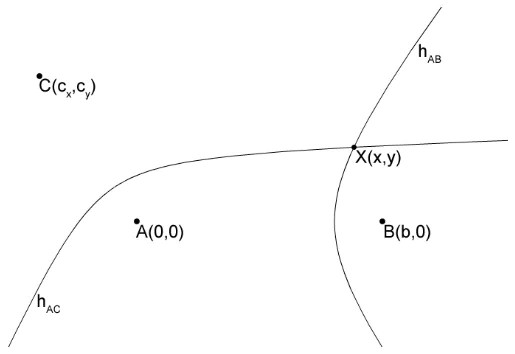

Figure III.9: Hyperbolas h_{AB} and h_{AC} given by their foci A, B and A, C, respectively.

such that for any point $X \in h_{AB}$, $|AX| - |BX| = R_{AB}$. Similarly, h_{AC} is defined by the foci A, C, and R_{AC}. Given the coordinates of A, B, C and the distances R_{AB}, R_{AC}, what are the intersection points of h_{AB}, h_{AC}?

Solution:

The following equations hold for the hyperbolas:

$$\begin{aligned} \sqrt{x^2+y^2} - \sqrt{(x-b)^2+y^2} = &\ R_{AB} \\ \sqrt{x^2+y^2} - \sqrt{(x-c_x)^2+(y-c_y)^2} = &\ R_{AC} \end{aligned} \qquad \text{(III.11)}$$

It is possible to solve this system of two equations analytically (see Appendix A) resulting in two solutions (x_1, y_1) and (x_2, y_2). Furthermore, cutting the plane along rays AB and AC, it can be shown that there is always at most one solution on each side. That is, the solution is unique if the location of the tracked node is confined in the area between the two rays.

Note that both Fig. III.9 and (III.11) assume that node A is at the origin and node B lies on the x-axis which is not true in general. This problem can be solved with coordinate transformation. We compute the translation matrix M_t that translates A to the origin, and the rotation matrix M_r that rotates the translated system such that the second coordinate of B be zero. We solve system of equations (III.11) for the coordinates of the tracked node in the translated coordinate system. Multiplying the resulting vector by $(M_t M_r)^{-1}$ gives the node's coordinates in the original coordinate system.

Another straightforward relaxation of the constraints of the above solution is to allow for 3D infrastructure node locations (but still localizing the tracked node in 2D). This approach is particularly practical when it is not possible (due to security reasons, or to avoid excessive multipath) to deploy the infrastructure nodes on the same plane where the tracked objects are expected to be moving. In a typical deployment scenario, where the tracked objects are people and the area of interest is an urban area, infrastructure nodes can be placed on rooftops and light poles.

3.8.2 Mobility related problems

Tracking algorithms approximate the continuous change of the location of the tracked objects with discrete location measurements. As we have discussed in Section III.6.2 the mobility of the tracked node becomes a major problem in radio interferometric ranging. The problem is that if the measured q-range changes by more than the wavelength during the ranging measurement, it becomes impossible to calculate the q-range from the phase offset measurements. The wavelength corresponds to 75 cm in our case which imposes severe limits on the maximum speed of the tracked objects.

For higher velocities, we propose to analyze the beat frequency of the interferometric signal to compensate for the changing q-range. It is a well known fact that moving objects measure frequencies that are Doppler shifted, depending on their speed and the direction of movement relative to the source of the measured signal. Interferometric ranging can actually measure the frequency of the beat signals with sufficient accuracy during the phase analysis of the signals. Not used in any of the distance computations for the stationary objects, the measured beat frequencies have been mere byproducts of the computation.

3.8.3 Doppler shifts in radio interferometry

The Doppler effects relate to the change of frequency of a wave that is perceived by an observer moving relative to the source of the waves. For radio waves traveling at the speed of light c, the relation between the frequency f of the emitted signal and the frequency shift Δf of the observed signal is

$$\frac{\Delta f}{f} \simeq -\frac{v_O}{c} \tag{III.12}$$

if the observer is moving directly away from the source with speed v_O, such that $v_O \ll c$. If the observer is moving towards the source, v_O should be taken negative. The Doppler effects are measurable because the frequencies we use are relatively large (400 MHz) and thus the observed frequency shifts are 1 Hz per 0.75 m/s velocity of the tracked node. Note that we measure the Doppler shifts of the high frequency radio signals indirectly, through the low frequency interference signals.

Let X be a tracked node participating in the target-as-receiver scenario. X proceeds by estimating its distance difference from two anchors A and B (t-range q_{ABX}) which transmit sine wave signals at slightly different frequencies. Let f_a, f_b be the frequencies of these radio signals, assuming $f_a < f_b$ without loss of generality. Let $\overrightarrow{v}$ be the velocity vector of X and let the anchors be stationary. If X was stationary, the observed frequency of the beat signal

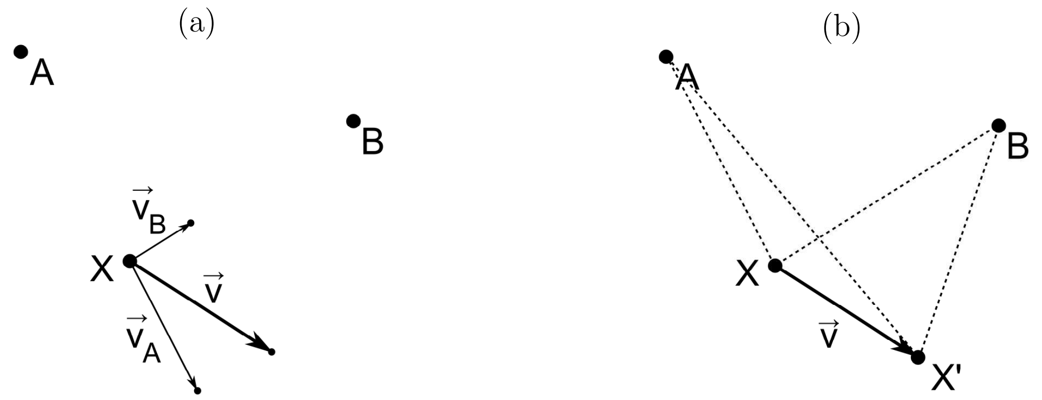

Figure III.10: Doppler effect in radio interferometric ranging.

would be $f_b - f_a$. However, if X is moving, the observed beat frequency depends on the speed of X relative to A and B.

For an arbitrary node A, let v_a be defined as the dot product of the unit vector in the direction of $\overrightarrow{AX}$ and the velocity vector $\overrightarrow{v}$:

$$v_a = \frac{\overrightarrow{AX}}{|\overrightarrow{AX}|} \cdot \overrightarrow{v}. \tag{III.13}$$

In other words, v_a is the length of the projection of vector $\overrightarrow{v}$ onto the line AX, with a negative sign if the projection vector points towards the node A and a positive sign otherwise (see Fig. III.10a).

The two signals transmitted by A and B interfere at X and the frequency of the resulting beat signal is:

$$\begin{aligned} f_{\text{beat}} = f_b + \Delta f_b - (f_a + \Delta f_a) &= \\ f_b - f_a - \frac{f_b}{c} v_b + \frac{f_a}{c} v_a. \end{aligned} \tag{III.14}$$

In our experiments $|f_a - f_b|$ is typically smaller than 1 kHz. Consequently, we can rewrite (III.14) using $\lambda = \frac{c}{f_a} \simeq \frac{c}{f_b}$:

$$f_{\text{beat}} = f_b - f_a + \frac{1}{\lambda}(v_a - v_b). \tag{III.15}$$

Note, that the frequency difference $f_b - f_a$ and the wavelength λ are known parameters[5] and the frequency of the beat signal f_{beat} can be measured. Therefore, we can calculate the term $v_a - v_b$ (the so called ***q-speed***) relating the actual speed of the moving object to A and

[5]Even though we may not be able to find f_a, f_b accurately enough, $f_b - f_a$ can be measured by a stationary anchor receiver C, for which $v_a - v_b = 0$. Consequently, f_{beat} measured by C exactly equals to $f_b - f_a$.

B. Even though we cannot compute v_a and v_b directly, the q-speed will become important in our later calculations.

3.8.4 Compensation of velocity error

The location of the tracked node is found at an intersection of multiple hyperbolas by measuring multiple t-ranges $q_{ABX} = d_{XA} - d_{XB}$. According to (III.4), phase offsets at multiple radio channels need to be measured to calculate the t-range q_{ABX}[6]. However, if X is moving, phase offsets are measured at different locations of X. In other words, the measured t-range q_{ABX}, which was assumed constant in (III.4), changes during the measurement. Therefore, the velocity of the node X needs to be incorporated in the calculations.

Consider Figure III.10b: we show here the case where the tracked node X measures phase offsets at two different carrier frequencies to disambiguate the t-range (in practice, more than two frequencies are needed). Points X and X' that correspond to the two measurement locations are potentially far away from each other, depending on the velocity $\overrightarrow{v}$ of node X and the ranging time. Assuming the velocity vector $\overrightarrow{v}$ does not change during the ranging measurement, the system of equations (III.4) can be rewritten using the t-range equation (III.10) as follows:

$$\begin{aligned} \Delta\gamma_i = &\quad d_{AX_i} - d_{BX_i} (\text{mod } \lambda_i) \\ X_i = &\quad X + i * \overrightarrow{v} * t_M. \end{aligned} \tag{III.16}$$

where t_M is the time required to make one phase offset measurement and where, by abuse of notation, we identified X with the position vector of the point X.

One problem of solving equations (III.16) is that the velocity vector $\overrightarrow{v}$ is unknown, adding greatly to the ambiguity of the ranging solution. Although the Doppler shift analysis provides some information on the velocity $\overrightarrow{v}$, it does not yield $\overrightarrow{v}$ directly.

Rather than solving equations (III.16) for the velocity $\overrightarrow{v}$ and the t-range $d_{AX} - d_{BX}$, we suggest to approximate d_{AX_i} and d_{BX_i} with $d_{AX} + v_a * i * t_M$ and $d_{BX} + v_b * i * t_M$ respectively. Using this approximation, we can express the measured distance differences as

$$d_{AX_i} - d_{BX_i} \simeq d_{AX} - d_{BX} + i * (v_a - v_b) * t_M.$$

[6]System of equations (III.4) describes q-range q_{ABCX} which can be easily converted to the t-range q_{ABX} using the known locations of anchors A,B, and C.

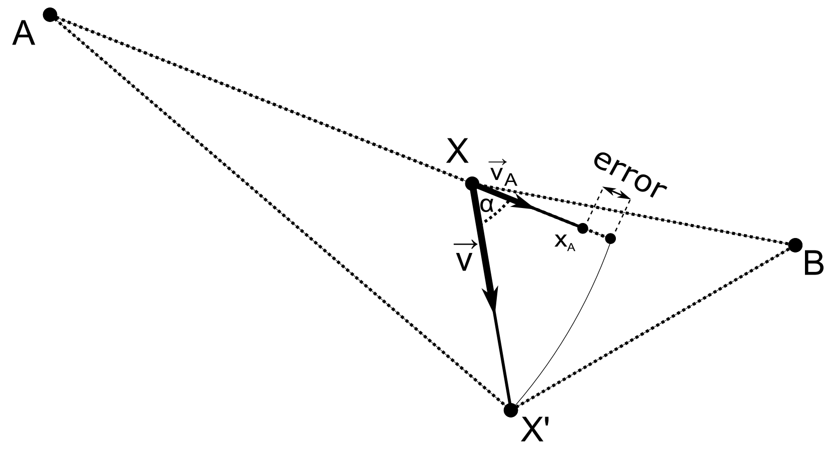

Figure III.11: Approximating movement errors. The tracked node node performs two ranging measurements, at locations X and X'. We approximate $d_{AX'}$ using the relative speed v_a of nodes X and A as $d_{AX'} \simeq d_{AX} + x_A$.

The q-speed $v_a - v_b$ can be calculated from Doppler shifts (III.15), allowing us to compute speed-compensated phase offsets $\Delta\nu_i$ from the measured offsets $\Delta\gamma_i$:

$$\Delta\nu_i = \Delta\gamma_i - (v_a - v_b) * t_M (\text{mod } \lambda_i).$$

Consequently, we rewrite (III.16) to

$$\Delta\nu_i \simeq \quad d_{AX} - d_{BX} (\text{mod } \lambda_i). \tag{III.17}$$

Equations (III.17) can be solved the same way as the original equations (III.4). However, the term $v_a - v_b$ only approximates the change of $d_{AX} - d_{BX}$ during the ranging measurement and we need to show that the error of this approximation is small.

Consider Figure III.11: node X computes its location from the beat signal generated by the anchors A and B. $\overrightarrow{v}$ is the velocity vector of the node X and $\overrightarrow{v_a}$ is projection of $\overrightarrow{v}$ onto AX. We analyze here the error of the approximation of the distance d_{AX}. The error of the d_{BX} approximation is analogous. Let t_M be the measurement time and $x_a = v_a * t_M$ be the distance that X travels along the direction $\overrightarrow{AX}$, then $d_{AX'}$ is approximated with $d_{AX} + x_a$. As seen in the figure, this approximation always underestimates the real value $d_{AX'}$ and is perfect if $\overrightarrow{v}$ and $\overrightarrow{v_a}$ are the same vectors. The error ε of the approximation can be expressed as $\varepsilon = d_{AX'} - d_{AX} - x_a$.

To find the maximum error of the approximation, we differentiated $\dfrac{\partial\varepsilon}{\partial\alpha}$ and found that the approximation error is maximal if $\cos\alpha = -\dfrac{d_{XX'}}{2d_{AX}}$, or, equivalently, if $d_{AX} = d_{AX'}$ (see

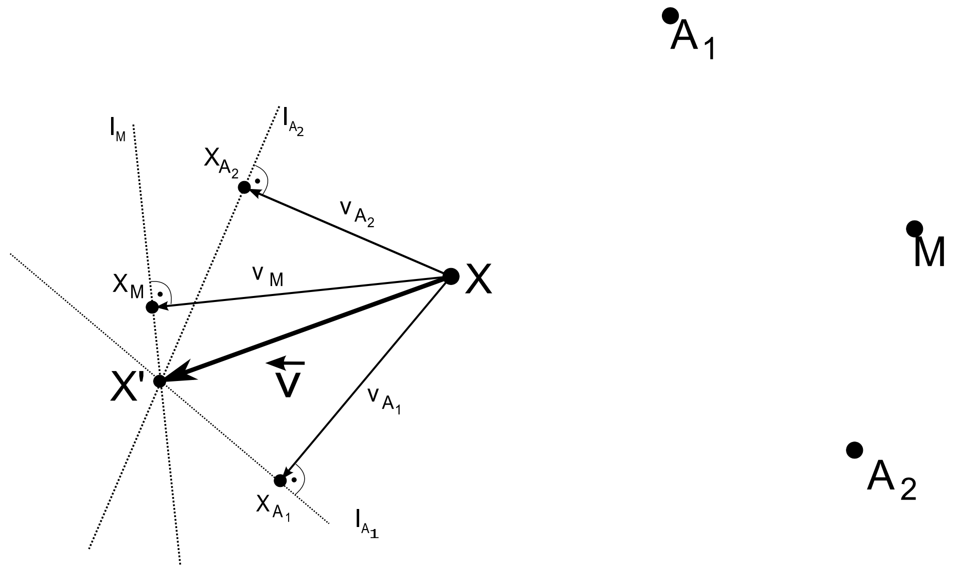

Figure III.12: Estimating the velocity vector v. A tracked node X measures two q-speeds from the Doppler shifts: $v_M - v_{A_1}$ and $v_M - v_{A_2}$. For a given estimate of v_M, the tracked node can calculate the lines l_M, l_{A_i} and estimate its velocity vector $v = XX'$.

Appendix B). Therefore, the maximum error can be calculated as

$$\varepsilon_{max} = |d_{AX} - d_{AX} - x_a| = |d_{XX'} \cos\alpha| = \frac{d_{XX'}^2}{2d_{AX}}. \tag{III.18}$$

In our experiments the measurement time t_M is less than 1 second. Therefore, $d_{XX'} = |\overrightarrow{v}|t_M$ is relatively small compared to d_{AX} and we conclude that the error of the approximation is relatively small. For example, if the tracked node moves 2 m/s, the ranging measurement takes 0.5 sec and the infrastructure nodes are 10 m away, the maximum error of our velocity compensation technique is 1/20m = 5cm.

3.8.5 Computing velocity vectors

The multiple t-ranges that are used to localize a mobile node can be selected in such a way, that a single infrastructure node participates in all t-ranges (see Section III.8.1). We have denoted this node by M (master node) and the nodes co-participating in the t-ranges by A_i , $i = 1 \dots k$ (assistant nodes). The q-speeds $v_M - v_{A_i}$ can be computed from the beat frequency of the interference signal using (III.15). We show that the velocity vector $\overrightarrow{v}$ of the tracked node can be computed utilizing q-speeds and the location of the tracked node X.

Consider Figure III.12 showing a master M and two assistant nodes A_1, A_2. The extension of our algorithm to more assistants is trivial. Let $\overrightarrow{v_M}$ and $\overrightarrow{v_{A_i}}$ be projections of

$\overrightarrow{v}$, onto XM and XA_i, respectively. Line l_M is defined as the line perpendicular to $\overrightarrow{v_M}$ which intersects the endpoint of vector $\overrightarrow{v_M}$ placed at the location of X. Lines l_{A_i} are defined analogously. It is easy to see that lines l_M and l_{A_i} intersect at a single point X' such that $\overrightarrow{v} = \overrightarrow{XX'}$.

We observe, that if we know one velocity projection, for example v_M, we can compute all other projections (v_{A_i} in this case) using the measured q-speeds and (III.15). From the q-speeds, we can find lines l_M, l_{A_i}, and their intersections points $P_{XY},\ X \neq Y \in \{M, A_i\}$.

Unfortunately, it is impossible for us to find the projected velocities v_M and v_{A_i} as only their differences are measured. However, it is easy to see that for incorrect v_M, the calculated v_{A_i} will all be incorrect and the intersections of the lines l_M and l_{A_i} will diverge from each other. This helps us to find the correct value of v_M. We use the maximum speed $v_{\max}$ as a parameter and iterate through all possible values of v_M from $[-v_{\max}, v_{\max}]$ interval with certain resolution. In each iteration, all intersection points P_{XY} are calculated and ideally, for the correct value of v_M, all these intersection points correspond to a single point which is X'.

More formally, in the i-th iteration step, let v^i be the current estimate of v_M. We define X^i as the center of gravity of all intersection points P^i_{XY} and a quality metrics $Q(X^i)$ of the solution X^i as the average distance of all points P^i_{XY} from their center of gravity X^i. The final result X' of our algorithm is then such X^i which minimizes $Q(X^i)$ for all v^i from $[-v_{\max}, v_{\max}]$. Finally, the velocity vector $\overrightarrow{v}$ is given by the vector $\overrightarrow{XX'}$.

3.8.6 Evaluation

Our experimental hardware platform is a Mica2 compatible ExScal mote [19], programmed using the nesC programming language [29] and the TinyOS operating system [42]. Our test environment is the empty Vanderbilt Football Stadium.

We have implemented ***mTrack*** tracking system capable of simultaneous tracking of multiple mobile nodes based on the target-as-receiver scenario. The analytical location solver (Section III.8.1), mobility error compensation (Section III.8.4), and velocity vector calculation (Section III.8.5) algorithms were implemented. mTrack system achieves a position fix in approximately 4.5 seconds for each of the tracked nodes which includes 0.5 second coordination time, 1.5 second ranging time, 2 seconds multihop routing time, and 0.5 second localization time. During the coordination phase, infrastructure nodes were assigned to participate in ranging and were synchronized with each other and with the tracked nodes using SyncEvent (see Section III.5.3). The nodes measured the phase and frequency of the beat signal during the ranging phase and routed the measured values to the base station during the routing phase. Finally, a PC computer calculated the locations and velocities of the

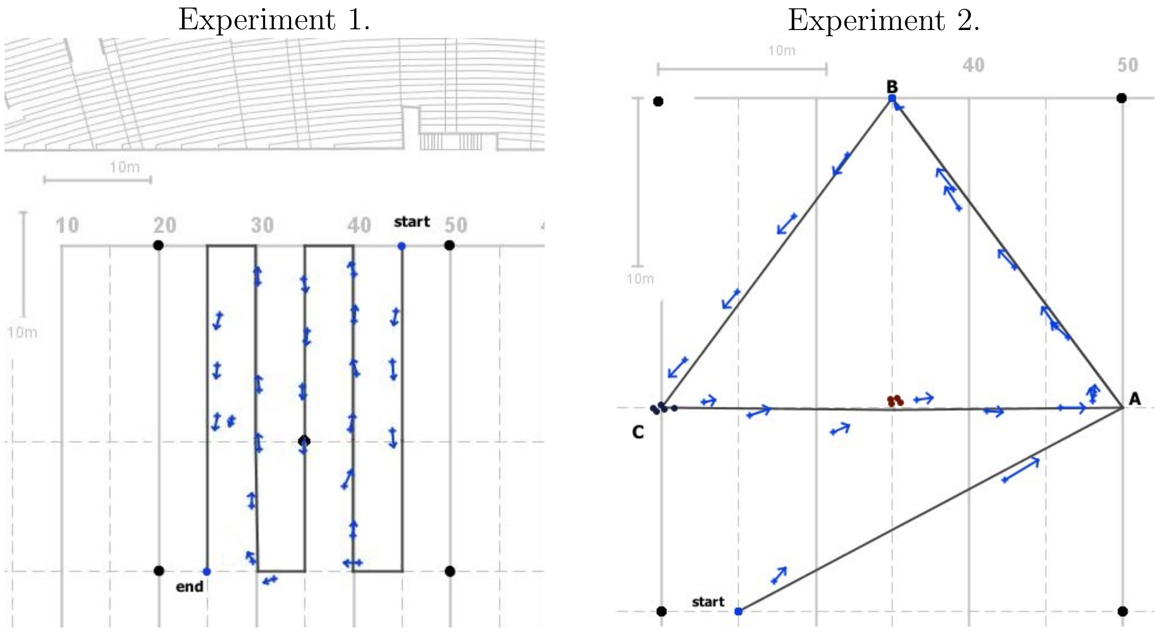

Figure III.13: Anchor nodes (black dots), mobile node track (black line), and location and velocity estimates (blue crosses and arrows) are shown. In Experiment 1, 5 anchor nodes are deployed and the mobile node starts at 45-th yard line. In Experiment 2, 4 anchor nodes are deployed. Mobile blue node follows ABCAB track, the black node is standing at C and the brown node is standing in the middle of AC.

tracked nodes during the localization phase, using the algorithms described in Section III.8.4 and Section III.8.5, respectively.

The location and velocity estimates were compared to the ground truth. However, measuring accurate ground truth locations of multiple simultaneously moving nodes was a difficult problem. We simplified this problem by predefining the tracks that the moving objects followed during the actual experiment. Each track consisted of a series of waypoints and the tracked objects moved between two consecutive waypoints at an approximately constant speed on a straight line. During the actual experiment, we recorded the times at which each of the tracked objects passed each of its predefined waypoints. This allowed us to compute the actual speed of each tracked node for each segment of its predefined track. We also recorded the times when the ranging measurements were taken, to be able determine the segment and interpolate the ground truth location of the tracked object on that segment for any given ranging measurement. Therefore, for any given ranging measurement, we were able to reconstruct the ground truth location and velocity of the tracked node.

It would seem that the 4.5 second position fix requirement would allow us to do rough accuracy analysis only, as the location of the tracked node would change significantly. However, the actual location related information is measured only during the ranging phase of the algorithm, the beginning of which can be determined precisely by analyzing the time

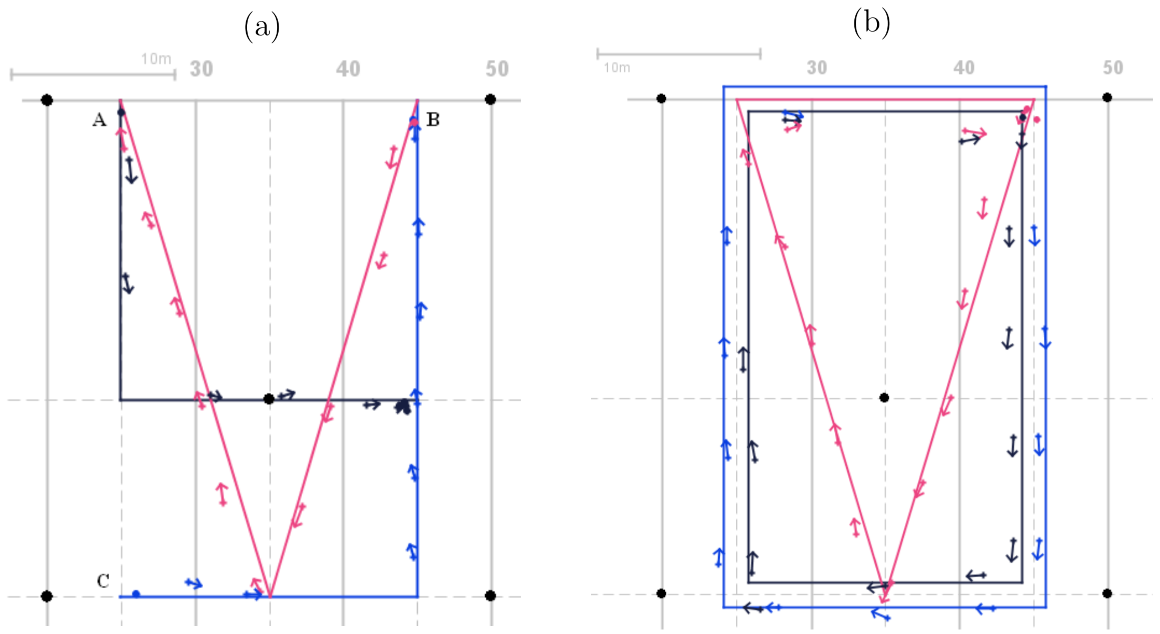

Figure III.14: Simultaneous tracking of multiple nodes. a) three different nodes are moving along 3 different tracks, starting at points A,B, and C. b) a person holds two motes in two hands, approximately 1.5 m apart and walks on the rectangular track, while a second person holds a single mote and walks on the triangular track.

synchronization messages. Recall that mTrack compensates for the velocity of the tracked node and the calculated location corresponds to the position of the tracked node at the beginning of the ranging phase (see Section III.8.4 and Fig. III.11). Therefore, we can determine the expected location of the tracked node accurately which allows for a meaningful accuracy analysis of the mTrack algorithm.

We conducted two experiments to illustrate tradeoffs between the accuracy and the infrastructure cost of our system. In our first experiment, we deployed five anchor nodes at known surveyed locations, covering an area of approximately 27.4×27.4 m and tracked two nodes simultaneously. We placed four anchors in the corners of this square and the fifth anchor close to the center. The actual setup can be seen in Fig. III.13 which shows the anchor node locations, the Vanderbilt stadium map, the track that a mobile node follows, and the calculated locations and velocity vectors. In our second experiment, we decreased the number of anchors to four and tracked three nodes simultaneously in the same area. We moved the mobile node at different speeds and in different directions, collecting approximately 100 data points for each experiment which took 10 minutes to complete. To be comprehensible, figures for both experiments show only a short subset of the whole dataset.

Note that in these experiments, only one node was mobile as establishing the ground truth for multiple nodes was difficult. Also, this allowed us to analyze the accuracy of our algorithm for both mobile and stationary nodes. The computational and radio bandwidth requirements of the nodes are identical, whether they are moving or not, as all of them run the same tracking algorithm.

Three types of errors were evaluated: localization, speed, and direction error with respect to the ground truth. We analyzed these errors separately for mobile and stationary nodes and calculated the average as well as 95-th percentile for all error types. Table III.1 shows the results achieved in both experiments *Mobile* and *Stationary* tables are showing the tracking accuracy only for mobile and only for stationary nodes, respectively.

Table III.1: Accuracy of tracking: tables show the average absolute error and the value of the 95%-th percentile in the parenthesis.

MOBILE:	**position (m)**	**Speed (m/s)**	**Angle (degree)**
Experiment 1	0.94 (1.82)	0.18 (0.45)	11.61 (30.88)
Experiment 2	1.64 (6.41)	0.2 (0.51)	13.29 (41.84)

STATIONARY:	**position (m)**	**Speed (m/s)**	**Angle (degree)**
Experiment 1	0.54 (1.1)	0.19 (0.67)	N/A
Experiment 2	0.83 (1.11)	0.2 (0.65)	N/A

As we can see, we can achieve 0.7 m better average accuracy using more anchor nodes. Using fewer anchor nodes, our ranging data is less accurate which results in higher chance of large localization errors (95-percentile is much larger). Further, the localization error is smaller for the stationary nodes. This is because we do not compensate the mobility related errors completely, but only use an approximation described in Section III.8.4 and because the ground truth is more accurate for stationary nodes. The error of the speed estimation in the stationary node case is due to the occasional errors in measuring the Doppler frequency. Finally, since the ground truth for the angle measurement in the stationary case is undefined, we show it as `N/A`.

We also wanted to test the case when multiple mobile nodes were tracked at the same time. However, due to the difficulties in recording the ground truth in this case, we only tested our system in a series of relatively short experiments. Even though the number of collected data was not statistically significant, we have observed errors similar to the mobile case in Table III.1. Two different experiments are shown in Fig. III.14.

3.9 Tracking with RF Doppler Shifts only (dTrack)

mTrack algorithm presented in the previous section provides a highly accurate location and velocity estimation technique utilizing low-cost and low-power hardware. Since multiple nodes can be tracked simultaneously and the required density of the infrastructure nodes is low, mTrack has a potential to provide cost effective tracking solution for WSN applications. One of the drawbacks of mTrack is its relatively large update rate (4 to 5 seconds) which limits its applicability in real time tracking. Recall that the large update rate was needed to improve the location estimation accuracy through generating the interference signal at a relatively large number of radio frequencies.

We propose to utilize similarly constructed interference signal, but instead of measuring its phase, we assume that the tracked node is mobile and measure the Doppler shifted frequency of the interference signal. The measurement process becomes simpler and faster when compared to mTrack:

(1) only the frequency of the interference signal is measured, allowing for simpler time-domain analysis of the interference signal,

(2) a single receiver computes the Doppler shift, allowing for a simpler data fusion. In contrast, a pair of receivers measured the range-yielding relative phase differences in mTrack, and

(3) the Doppler shifts are measured at a single carrier frequency, whereas as much as 21 different frequencies are required in mTrack to obtain highly accurate ranging data from phase measurements.

Further advantage of measuring the Doppler shifts is that it is less susceptible to the errors introduced by multipath propagation. The main reason is that the reflected radio signal has the same frequency as the original signal. Therefore, additional Doppler shifts can only be introduced by multipath between the mobile tracked node and a given receiver. In contrast, phase of the radio signal can be distorted by multipath propagation between all four transmitter-receiver pairs.

We have designed the *Doppler-based tracking service (dTrack)* that estimates both the *velocity* and the *location* of the tracked nodes simultaneously, utilizing the Doppler-shifted frequency measurements. The main challenge that dTrack has to overcome is that it cannot rely on the accurate ranging information which was obtained from the phase offsets measurements in mTrack. The location and velocity of the tracked nodes have to be estimated indirectly, from the Doppler shifts, or in other words, relative target velocities measured at the infrastructure nodes. Thus, the localization problem becomes significantly harder to solve than in mTrack. We have utilized an Extended Kalman Filter (EKF) technique to accurately track mobile nodes under the constant velocity assumption because it can eliminate the effects of noisy measurements on the tracking accuracy. Further, we have extended

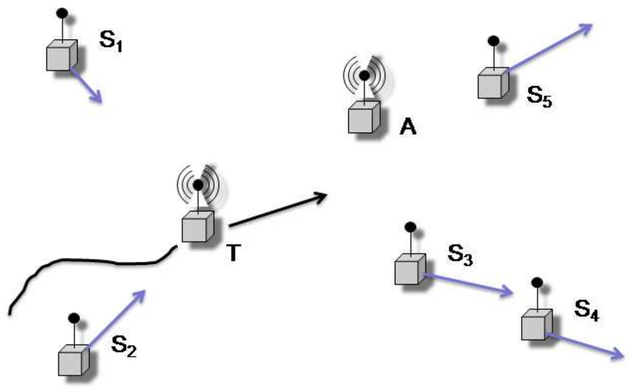

Figure III.15: Tracked node T is moving through the sensor field while one infrastructure node A is transmitting at the same time. Five other infrastructure nodes (S_i) calculate the relative speed of T (blue arrows) from the measured Doppler shifted frequencies.

EKF with Constrained Non-linear Least Squares (CNLS) technique to get a good tracking accuracy if the tracked object is maneuvering (and thus the constant velocity assumption does not hold). The CNLS-EKF algorithm is computationally efficient and provides good tracking accuracy in both maneuvering and non-maneuvering cases.

3.9.1 Approach

dTrack follows the general structure of many of the existing algorithms [8, 78, 31]:

Coordination phase: infrastructure nodes are notified to participate in tracking of a node in a certain region. Both the timeframes and the local coordinate systems of the participating nodes need to be synchronized to enable the data fusion of spatially and temporally distributed measurements.

Measurement phase: ranging measurements that provide information on the location, bearing, and/or speed of the tracked object are collected. The low level data is stored locally or handed off to a different node, for example, a higher level data fusion node.

Tracking phase: we use non-linear optimization and filtering techniques to smooth out the measurement noise by combining the ranging data measured at multiple infrastructure nodes. The movement of the tracked node can be predicted and the infrastructure nodes participating in the tracking can be activated or deactivated accordingly.

3.9.2 Measuring Doppler shifts

In Section III.8.3, we have shown that radio interferometry can be extended to measure the Doppler shifts of the interference signal to improve the accuracy of the mTrack algorithm in the mobile node case as well as to estimate the velocity of the mobile node. dTrack utilizes the target-as-transmitter case shown in Fig. III.15: the tracked node T transmits an unmodulated sine wave at frequency f_t, an infrastructure node A transmits an unmodulated sine wave at frequency f_a, such that $f_t > f_a$. The two sine waves interfere with each other and create a signal with the envelope frequency of $f_t - f_a$. The interference signal is measured by a number of infrastructure nodes S_i. Since T moves relative to the infrastructure nodes, Doppler shifted frequencies will be observed. The signal transmitted at f_t will be Doppler shifted by Δf_t^i at each node S_i where the magnitude of Δf_t^i depends on the relative speed of T and S_i. The relative speed of the infrastructure nodes is 0, therefore, the signal transmitted by A is not Doppler shifted. It follows that the measured envelope frequency f_i of the interference signal at node S_i is given by

$$f_i = f_t - f_a + \Delta f_t^i. \tag{III.19}$$

Consequently, using (III.19), we can calculate the Doppler shift at node S_i using the measured frequency f_i, if the transmitted frequencies f_t and f_a are known.

We apply the Doppler equation (III.15) to the dTrack beat frequency equation (III.19). Since node A is stationary, only the Doppler shift in the frequency f_t will be observed. Consequently, the node S_i observes the interference signal with frequency f_i given as follows (using $\widehat{f} = f_t - f_a$):

$$f_i \quad = \quad \widehat{f} - v_i/\lambda_t, \tag{III.20}$$

where v_i was defined in (III.13) as the relative speed of the tracked node with respect to the infrastructure node S_i and λ_t is the wavelength of the radio signal transmitted by the tracked node.

Equation (III.20) allows us to compute the interference frequency observed at each of the infrastructure nodes, if we know the difference of the two transmitted frequencies $\widehat{f}$, and the location and velocity of the tracked node T. Note that estimating the frequency difference $\widehat{f}$ with sufficient accuracy becomes a problem when using low-cost radio transceivers (see Section III.9.10). Consequently, we have to treat $\widehat{f}$ as an unknown parameter in our tracking algorithm.

3.9.3 Tracking as optimization problem

Tracking is a higher level algorithm that utilizes the Doppler shifts measured by multiple infrastructure nodes to calculate the location and the velocity of a tracked node. We model

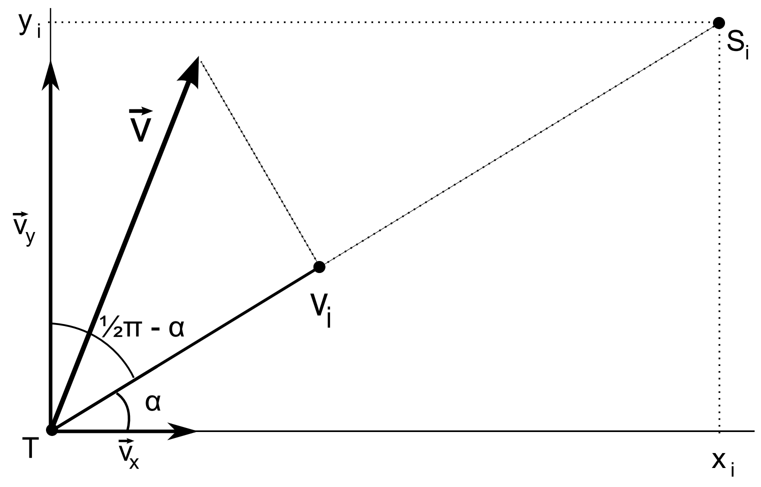

Figure III.16: Tracked node T with velocity $\overrightarrow{v}$ transmits a signal. Sensor S_i measures the Doppler shift of the signal which depends on v_i, the relative speed of T and S_i.

the tracking problem as a non-linear optimization problem and use non-linear least squares (NLS) optimization method to solve it.

Parameters that need to be estimated by the optimization are the location (x, y) and the velocity vector $\overrightarrow{v} = (v_x, v_y)$ of the tracked node. Also, we do not know the transmitted frequencies f_t and f_a with sufficient accuracy (see Section III.9.10), hence their difference $\widehat{f}$ becomes one of our parameters. Consequently, we define the parameter vector $\mathbf{x}$ as

$$\mathbf{x} = (x, y, v_x, v_y, \widehat{f})^{\mathsf{T}}.$$

Assuming that n infrastructure nodes measure the Doppler shifted radio signal, we have n frequency observations (f_i) that are the constraints for the optimization. We define the observation vector $\mathbf{c}$ as

$$\mathbf{c} = (f_1, f_2, \ldots, f_n)^{\mathsf{T}}.$$

The parameter vector $\mathbf{x}$ and the observation vector $\mathbf{c}$ are related to each other. We formalize this relation through a function $F : \mathcal{R}^5 \to \mathcal{R}^n$, such that

$$F(\mathbf{x}) = \mathbf{c}.$$

The function F is a vector function consisting of n functions $F_i : \mathcal{R}^5 \to \mathcal{R}$, each of them calculating the Doppler shifted interference frequency f_i measured at an infrastructure node S_i. Using (III.20), $F_i(\mathbf{x})$ can be expressed as:

$$F_i(\mathbf{x}) = \widehat{f} - v_i/\lambda_t.$$

Consider Fig. III.16. We use the distributive property of the dot product, and identities $\overrightarrow{v} = \overrightarrow{v_x} + \overrightarrow{v_y}$, $\sin\alpha = \cos(\frac{\pi}{2} - \alpha)$, and $|\overrightarrow{u_i}| = 1$ (the length of a unit vector is 1) to express v_i:

$$\begin{aligned} v_i &= \overrightarrow{u_i} \cdot \overrightarrow{v} = \overrightarrow{u_i} \cdot \overrightarrow{v_x} + \overrightarrow{u_i} \cdot \overrightarrow{v_y} \\ &= |\overrightarrow{u_i}|.|\overrightarrow{v_x}| \cos\alpha + |\overrightarrow{u_i}|.|\overrightarrow{v_y}| \cos(\frac{\pi}{2} - \alpha) \\ &= |\overrightarrow{v_x}| \cos\alpha + |\overrightarrow{v_y}| \sin\alpha \end{aligned}$$

Finally, assuming that the coordinates of S_i are (x_i, y_i), we express $\sin\alpha$ and $\cos\alpha$ using the coordinates of T and S_i and get the following equation for F_i:

$$F_i(\mathbf{x}) \quad = \quad \widehat{f} - \frac{1}{\lambda_t} \frac{v_x(x_i - x) + v_y(y_i - y)}{\sqrt{(x_i - x)^2 + (y_i - y)^2}} \tag{III.21}$$

We estimate the parameters $\mathbf{x}$ for a given observation vector $\mathbf{c}$ can be done by finding $\mathbf{x} \in \mathcal{R}^5$ such that $\|F(\mathbf{x}) - \mathbf{c}\|$ is minimized. Note that the components of our objective function F are non-linear functions, requiring us to use non-linear optimization methods.

3.9.4 Non-linear Least Squares (NLS)

NLS is an iterative method for non-linear optimization with an advantage of being computationally efficient as it requires small number of operations per iteration.

Given the function $F : \mathcal{R}^5 \to \mathcal{R}^n$ and the observation vector $\mathbf{c}$, the NLS algorithm optimizes $\|F(\mathbf{x}) - \mathbf{c}\|$ by minimizing the objective function $\mathcal{F} : \mathcal{R}^5 \to \mathcal{R}$, defined as

$$\mathcal{F}(\mathbf{x}) \quad = \quad \frac{1}{2} \sum_{i=1}^{n} (F_i(\mathbf{x}) - c_i)^2 = \frac{1}{2} (F(\mathbf{x}) - \mathbf{c})^\mathsf{T} (F(\mathbf{x}) - \mathbf{c}). \tag{III.22}$$

An example of the function $\mathcal{F}$, calculated based on actual experimental data measured by eight infrastructure nodes deployed on our campus, is plotted in Fig. III.17.

Function $F(\mathbf{x}) - \mathbf{c}$ is linearized by Taylor expansion in the neighborhood of $\mathbf{x}$ as

$$F(\mathbf{x} + \Delta) - \mathbf{c} \simeq l(\Delta) = F(\mathbf{x}) - \mathbf{c} + J(\mathbf{x})\Delta$$

where $J \in \mathcal{R}^{n \times 5}$ is the Jacobian of $F(\mathbf{x}) - \mathbf{c}$, calculated in Appendix C. Similarly, using (III.22), $\mathcal{F}$ is linearized as

$$\mathcal{F}(\mathbf{x} + \Delta) \simeq L(\Delta) = \frac{1}{2} l(\Delta)^\mathsf{T} l(\Delta).$$

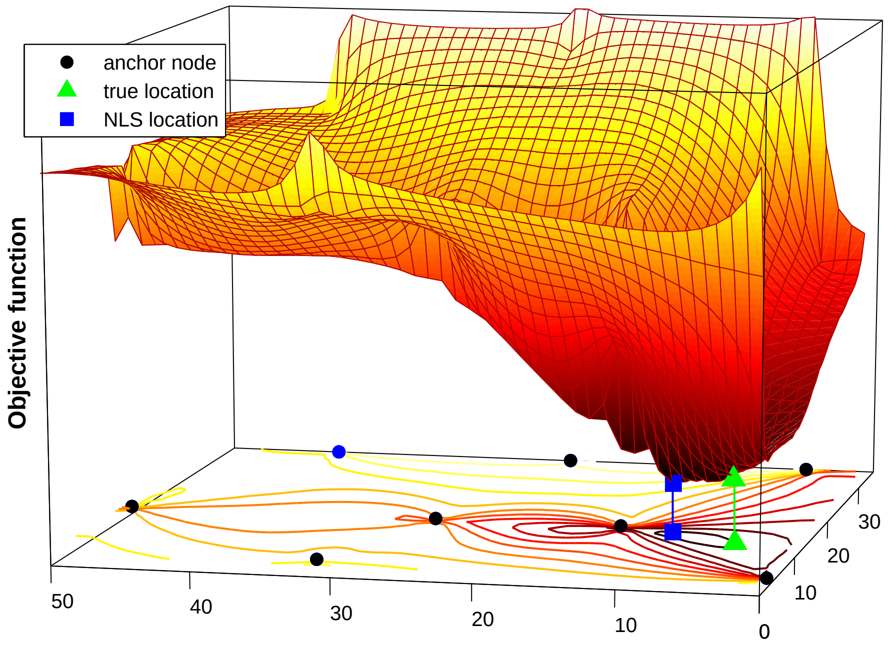

Figure III.17: The objective function $\mathcal{F}$ minimizes the least-square error in the observation vector. Due to the measurement error, the global minimum of $\mathcal{F}$ is not at the true location of the tracked node. For illustration, both true and NLS locations are projected to the xy-plane.

The formula for the gradient of L can be derived:

$$L'(\Delta) = J(\mathbf{x})^{\mathsf{T}} F(\mathbf{x} - \mathbf{c}) + J(\mathbf{x})^{\mathsf{T}} J(\mathbf{x})\Delta,$$

which, after letting $L'(\Delta) = 0$, gives us the solution for the local minimizer of $\mathcal{F}$ around $\mathbf{x}$

$$\Delta_{\text{min}} = (J(\mathbf{x})^{\mathsf{T}} J(\mathbf{x}))^{-1} J(\mathbf{x})^{\mathsf{T}} (\mathbf{c} - F(\mathbf{x})). \tag{III.23}$$

Full derivation of this equation can be found in [66].

Consequently, the least squares method iterates through a series of parameter vectors $\mathbf{x_0}, \mathbf{x_1}, \ldots,$ from a starting point $\mathbf{x_0}$ the following way, assuming the i-th vector $\mathbf{x_i}$ is given:

(1) calculate the Jacobian $J_i = J(\mathbf{x_i})$,

(2) calculate the local minimizer Δ_i from (III.23), and

(3) calculate $\mathbf{x_{i+1}} = \mathbf{x_i} + \alpha\Delta_i$, where α is the step length influencing the convergence of the method. The algorithm stops if Δ_i is small.

3.9.5 Constrained Non-linear Least Squares (CNLS)

In tracking, we are often able to constrain the area where the tracked node is located. Therefore, applying CNLS optimization may yield more accurate results. One way to solve the constrained optimization is to modify the objective function by adding a barrier function to it, introducing zero penalty inside the region of interest and positive penalty outside of it.

We want to constrain the tracked node location[7] to a disk centered at (x_0, y_0), with radius r. A logarithmic barrier function $\mathbf{b}(\mathbf{x})$ can be then defined

$$\mathbf{b}(\mathbf{x}) = -\log(r - \sqrt{(x - x_0)^2 + (y - y_0)^2}),$$

$\mathbf{x} = (x, y, v_x, v_y, \widehat{f})$ is the parameter vector. Note that as $\sqrt{(x - x_0)^2 + (y - y_0)^2}$ approaches r, the logarithm goes to $-\infty$ and the penalty function goes to ∞.

The CNLS algorithm works the same way as the NLS algorithm, except it optimizes the function $\mathcal{F}(\mathbf{x}) + \mathbf{b}(\mathbf{x})$. The derivatives used in the NLS algorithm need to be adjusted by adding the term $B(\mathbf{x}) = \dfrac{\partial \mathbf{b}(\mathbf{x})}{\partial \mathbf{x}}$ to $L'(\Delta)$. Consequently, (III.23) becomes

$$\Delta_{\min} = (J(\mathbf{x})^{\mathsf{T}} J(\mathbf{x}))^{-1} [J(\mathbf{x})^{\mathsf{T}} (\mathbf{c} - F(\mathbf{x})) + B(\mathbf{x})]. \tag{III.24}$$

The Jacobian $B(\mathbf{x})$ of our barrier function can be found in Appendix C.

3.9.6 Problems with CNLS optimization

Non-linear least squares optimization may fail depending on the starting point $\mathbf{x_0}$ and the measurement errors that corrupt the observation vector $\mathbf{c}$. This is because the solution will converge to a local minimum of the objective function, or because the observations are insufficient to determine all the parameters.

We confirmed experimentally that our objective function $\mathcal{F}$ is susceptible to these problems. We placed eight infrastructure nodes in a 30×50 m area on our campus and measured the Doppler shifted frequency of the transmitted signal. In Fig. III.17, we show the locations of the infrastructure nodes as black dots and the location of the transmitter as a gray triangle. Recall that $\mathbf{x}$ consists of five parameters x, y, v_x, v_y, and $\widehat{f}$. The function plotted in the figure is obtained by finding the minimum value of $\mathcal{F}(\mathbf{x})$ for the fixed coordinates (x, y) (i.e., finding the best fit for the three remaining parameters). Fig. III.18 shows the best-fit velocities found at a given location.

[7]We found that constraining the values of parameters v_x, v_y, and $\widehat{f}$ was not necessary because their errors were insignificant.

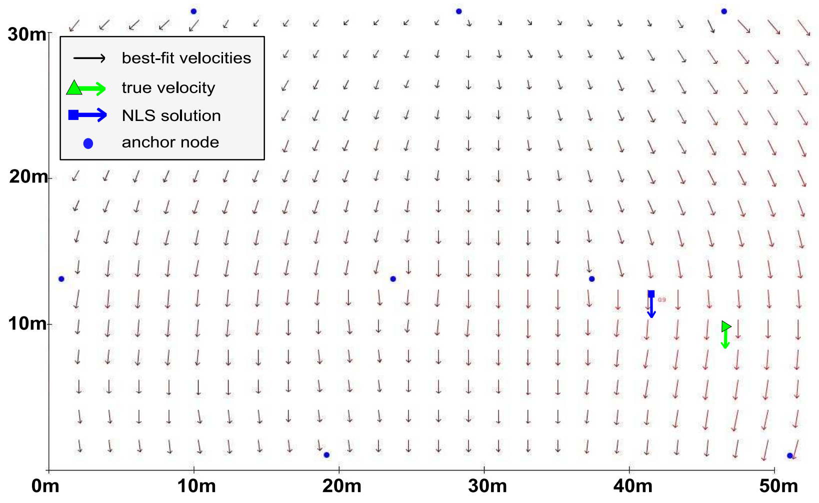

Figure III.18: The best-fit velocities of the tracked node that minimize function $\mathcal{F}$ in Fig. III.17 are shown.

We make two observations: (1) function $\mathcal{F}$ indeed contains multiple local minima close to the location of the tracked node, thus the constrained NLS algorithm is required, and (2) the global minimum of $\mathcal{F}$ (the gray square) is 5.6 m away from the true location of the transmitter (the gray triangle), therefore, all optimization techniques introduce a large localization error in this case. Since the optimization methods fail to find the correct location of the tracked node in certain cases, we need to find alternate solutions to our tracking problem.

On the positive side, the objective function is relatively smooth and converges fast to the general area where the tracked node is located. Also, the estimated velocity of the tracked node, as shown in Fig. III.18, is accurate in a relatively large area around the true location of the tracked node. This allows us to use the velocities calculated with CNLS algorithm with higher confidence than the locations.

3.9.7 Extended Kalman Filter (EKF)

Noise corrupted observations may prevent us from solving the tracking problem with sufficient accuracy, as shown in the previous section. Therefore, we resort to state estimation techniques which model the dynamics of the tracked node, estimate the state of the node based on its dynamics, and update the state based on the new observations. A Kalman filter (KF) is a widely used method for the state estimation of dynamic systems using a set of noisy measurements. The current state $\mathbf{x_k}$ of the system is calculated recursively from the previous state $\mathbf{x_{k-1}}$ and a new observation vector $\mathbf{c_k}$. An error covariance matrix $\mathbf{P}$, a measure of the accuracy of the estimated state $\mathbf{x}$, is calculated along with $\mathbf{x}$.

An important step in designing a KF is modeling the dynamics of the observed system. The KF framework requires that the system state $\mathbf{x_k}$ evolves according to

$$\mathbf{x_k} = F(\mathbf{x_{k-1}}) + \mathbf{w_k},$$

where $\mathbf{x_{k-1}}$ is the previous state, F models the system dynamics, and $\mathbf{w_k}$ is the process noise with covariance $\mathbf{Q}$.

The KF framework also defines the model of the measurement process which is non-linear with respect to $\mathbf{x}$ in our case. An Extended Kalman Filter (EKF) can be applied which essentially linearizes the measurement function by calculating its Jacobian around the current state estimate. The observation $\mathbf{c_k}$ of the current state is modeled as

$$\mathbf{c_k} = H(\mathbf{x_k}) + \mathbf{v_k},$$

where H is the non-linear measurement function and $\mathbf{v_k}$ is normally distributed measurement noise with covariance $\mathbf{R}$.

$\mathbf{x_k}$ and $\mathbf{P_k}$ constitute the state of the EKF. This state is updated in two phases: (1) a time prediction phase during which the current state is estimated based on the previous state, and (2) a measurement update phase during which the new measurements are used to refine the predicted state.

KF prediction phase. Given the previous state of the KF $(\mathbf{x_{k-1}}, \mathbf{P_{k-1}})$, the predicted state $(\mathbf{x_k^-}, \mathbf{P_k^-})$ is

$$\begin{aligned} \mathbf{x_k^-} &= \mathbf{F}\mathbf{x_{k-1}} \\ \mathbf{P_k^-} &= \mathbf{F}\mathbf{P_{k-1}}\mathbf{F}^\mathsf{T} + \mathbf{Q}, \end{aligned} \tag{III.25}$$

where $\mathbf{Q}$ is the covariance matrix of the process noise and $\mathbf{F}$ is the function (matrix) that models the system dynamics.

EKF update phase. The predicted state $(\mathbf{x_k^-}, \mathbf{P_k^-})$ is updated with $\mathbf{c_k}$ and the new state $(\mathbf{x_k}, \mathbf{P_k})$ is

$$\begin{aligned} \mathbf{K_k} &= \mathbf{P_k^-}\mathbf{J_k}^\mathsf{T}(\mathbf{J_k}\mathbf{P_k^-}\mathbf{J_k}^\mathsf{T} + \mathbf{R})^{-1} \\ \mathbf{x_k} &= \mathbf{x_k^-} + \mathbf{K_k}(\mathbf{c_k} - H(\mathbf{x_k^-})) \\ \mathbf{P_k} &= (\mathbf{I} - \mathbf{K_k}\mathbf{J_k})\mathbf{P_k^-}, \end{aligned} \tag{III.26}$$

where $\mathbf{R}$ is the measurement noise, $\mathbf{K_k}$ is Kalman gain, and $\mathbf{J_k}$ is the Jacobian matrix of H in $\mathbf{x_{k-1}}$.

Model of our system: The system state $\mathbf{x} = (x, y, v_x, v_y, \widehat{f})$ and the measurement vector $\mathbf{c} = (f_1, \ldots, f_n)$ are equivalent to the parameter and observation vectors defined in Section III.9.3, respectively. Both the process noise $\mathbf{w}$ and the measurement noise $\mathbf{v}$ are assumed to have "white noise" properties with zero mean and their covariant matrices will be determined in the implementation and evaluation sections.

The state transition matrix F of our linear state space is

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & t & 0 & 0 \\ 0 & 1 & 0 & t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where t is the time elapsed from the previous state.

The non-linear observation function H is defined using the vector function $\mathbf{f}$ defined in (III.21):

$$H_i(\mathbf{x}) \quad = \quad F_i(\mathbf{x})$$

and its Jacobian $\mathbf{J}$ can be found in Appendix C.

3.9.8 Problems with the EKF

We applied the EKF to our tracking problem and found that the filter tracks the mobile nodes accurately, removing the effects of the measurement noise. However, EKF works well only under the assumption of constant velocity of the tracked node. Situations, when the tracked node changes its direction significantly, for example by 180°, are potentially the most severe. We illustrate this problem in Fig.III.19: we deployed a number of infrastructure nodes (black dots) on campus and moved the tracked node with a constant speed. After some time, we changed both the direction and the speed of the tracked node and observed how the EKF handles this situation. The track that the node followed consisted of two ~ 30 m segments. As we can see, the filter requires six measurement rounds to converge back to the true location, introducing significant errors in the location estimates in this transition period.

The divergence of the EKF in tracking maneuvering nodes is a well known problem. A simple solution is to increase the process noise covariance $\mathbf{Q}$, assigning more weight to the measurements than to the prediction model. This, however, degrades the overall tracking accuracy as the effects of the measurement noise are more significant. Other techniques

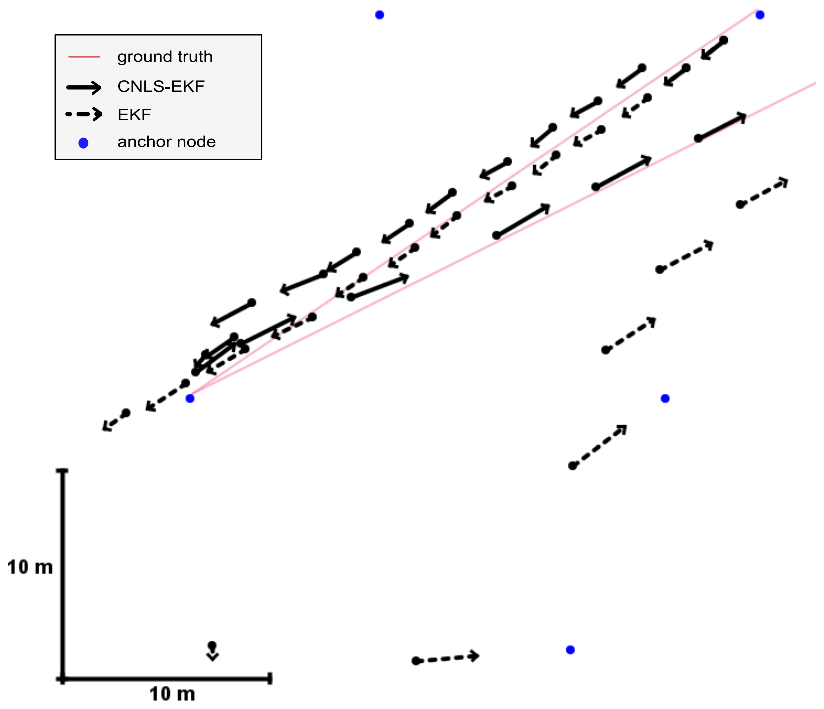

Figure III.19: The EKF fails if the tracked node makes sudden maneuvers. The CNLS-EKF outperforms the EKF in the maneuvering case, while keeping good performance in the non-maneuvering case.

propose to simultaneously use multiple Kalman filters which are set up with different models of the tracked node dynamics. Also, the EKF update algorithm can be executed iteratively, to better approximate the error function locally. The disadvantage of these techniques is their higher computational complexity, as the KF prediction and update need to be executed multiple times per observation.

3.9.9 CNLS-EKF algorithm for the maneuvering target

We propose to combine the EKF and the CNLS techniques in a way that significantly improves the tracking accuracy in the maneuvering case, while maintaining the good performance of the EKF in the non-maneuvering case. The combined algorithm is referred to as the CNLS-EKF algorithm. The main motivation for choosing the CNLS method is its computational efficiency and fast convergence, given a good initial estimate of the parameters.

The CNLS-EKF algorithm proceeds in three steps:

(1) given a new observation vector $\mathbf{c}$, calculate the new EKF state $\mathcal{S} = (\mathbf{x}, \mathbf{P})$ using (III.25) and (III.26),

(2) if a maneuver is detected, find a new system state $\mathbf{x}^*$ by running the CNLS optimization initiated at $\mathbf{x}$, and

(3) use the new system state $\mathbf{x}^*$ to update the EKF state $\mathcal{S}$.

Maneuver detection algorithm:
In our experience, maneuvers can be detected reliably when the direction and the speed of the tracked node change significantly from their last estimates. This is because the velocity estimates are relatively robust to both the measurement errors and the errors in the predicted location (see Fig. III.18).
CNLS-based EKF update:
The EKF state $\mathcal{S} = (\mathbf{x}, \mathbf{P})$ is updated by running the linear KF update algorithm using CNLS-computed system state $\mathbf{x}^*$ as the observation vector. The measurement matrix H^* used in the linear KF update is simply an identity matrix I_5. The covariance matrix R^* determines how much the CNLS solution influences the state $\mathcal{S}$ during the linear KF update. Since the CNLS optimization was shown to be sensitive to measurement errors, we need to limit its influence in the non-maneuvering case. Therefore, we define the covariance matrix R^* with exponentially increasing values over time: the more time has passed since the maneuver, the smaller the influence of $\mathbf{x}^*$ on the state $\mathcal{S}$. If the time elapsed from the last detected maneuver is Δt seconds, we define R^* as

$$\mathbf{R}^* = \rho \begin{bmatrix} 5^{\Delta t} & 0 & 0 & 0 & 0 \\ 0 & 5^{\Delta t} & 0 & 0 & 0 \\ 0 & 0 & 2^{\Delta t} & 0 & 0 \\ 0 & 0 & 0 & 2^{\Delta t} & 0 \\ 0 & 0 & 0 & 0 & 2^{\Delta t} \end{bmatrix}.$$

The base of the exponential function is higher for the location coordinates than the velocity coordinates because the velocity estimates are less prone to the measurement errors and the initial state choice (see Fig. III.18). We keep the error of the interference frequency $\widehat{f}$ small, because it does not change significantly between consecutive measurements. The scaling factor ρ is found experimentally.

The improvement of the CNLS-EKF algorithm over regular EKF can be seen in Fig. III.19. The tracking accuracy can improve by as much as 36% (see Section III.9.11).

3.9.10 Implementation

Similarly to mTrack, we have implemented dTrack using the TinyOS operating system [63] and ExScal nodes [19]. In the current proof-of-concept implementation, the CNLS-EKF algorithm runs on a PC-class device, utilizing the Doppler shifts measured by the sensor nodes.

Interference signal: The low-cost radio transceiver that we use does not allow for setting the transmission frequency accurately. Ten parts-per-million (ppm) clock oscillator errors are common, which correspond to ± 4 kHz frequency errors at the 400 MHz operating frequency of our radios. This is a problem, because we need to know the actual radio frequencies f_t and f_a accurately to measure the Doppler shifts in the observed radio signal (see (III.19)). For this reason, we have to treat $\widehat{f} = f_t - f_a$ as one of the unknown parameters in our algorithm. Note that the accurate value of the wavelength $\lambda_t = c/f_t$ is also required in our equations to calculate the relative velocities from the Doppler frequency shifts (see (III.20)). However, it is easily seen that 4 kHz error corresponds to a small wavelength error at 400 MHz and the use of the approximate wavelength is sufficient.

Making sure that the frequency difference of two radio signals is within the given operating range is an easier problem than calibrating both nodes to transmit at accurate frequencies and it is done by the tuning algorithm described in Section III.5.3.

Measuring Doppler shifts: The tuning algorithm will maintain the beat frequency in a pre-defined operating range, such as $300 - 400$ Hz. We have implemented a simple time-domain signal analyzer which runs online at a 9 kHz sampling. We find all peaks in the signal filtered with a moving average filter. A period is defined as the number of samples between any two consecutive peaks. Since we know the expected period (i.e., 22–30 samples given the 9 kHz sampling and the 300–400 Hz expected beat frequency), we do additional filtering by removing the periods that are outside of the expected interval. Consequently, the frequency is computed as reciprocal of the average period p. This algorithm runs online and does not require a post-processing phase.

The accuracy of the frequency measurement depends on the size of the fixed window in which we observe the signal. Alternatively, one can perform the frequency measurement multiple times for smaller windows and average these results, improving the accuracy and error resilience in case part of the signal was corrupted. Therefore, the accuracy of our algorithm can be improved at the price of longer measurement time.

To find the trade-off between the time required and the accuracy achieved, we evaluated our algorithm in a series of experiments. We placed ten nodes at different locations, keeping them stationary for the duration of the experiment. Two nodes were chosen to be the transmitters and the remaining eight nodes measured the beat frequency and reported it back to the base station. We varied two parameters: (1) the size of the sampling window (255,450,650, and 1050 samples) and (2) the number of frequency measurements (1,2,...,6). Since none of the nodes was moving, all receivers were measuring the same frequency. The standard deviation of the frequencies recorded by different receivers gives us an estimate of the accuracy of our algorithm. We have carried out approximately 1000 measurements for each pair of parameters and calculated the average standard deviation as well as the corresponding measurement length.

Based on these experiments, we have chosen to have 450 samples in our observation window and repeated the measurement 6 times. For these parameters, the overall measurement time is 0.4 sec and the standard deviation of the observed measurements is 0.21 Hz. The 0.21 Hz error in the Doppler shift corresponds to 0.15 m/s error in the target speed in the 430 MHz band. We have also studied how the interference frequency changes over time, observing 1 Hz standard deviation of the frequencies measured a few seconds apart. Both the variance of the measurement process and the variance of the interference frequency are important design parameters for the EKF algorithm.

3.9.11 Experimental evaluation

A number of ExScal motes [19] were deployed that served both as the stationary infrastructure nodes and the mobile tracked node. An online version of our tracking algorithm was running while a person was walking and running with the tracked node.

Setup description: We have utilized eight infrastructure nodes that measured the Doppler effect and deployed them in a 50×30 m area (see Fig. III.20). Since two nodes are required to transmit in our approach, one more infrastructure node was used to co-transmit with the tracked node. We measured the ground truth locations of the infrastructure nodes with an estimated error of 0.5 m. The resulting network had two-hop diameter for the duration of the experiment. We have estimated the ground truth location and velocity of the tracked nodes analogously to the mTrack evaluation: mobile nodes followed predefined tracks and we recorded the ranging measurement times to be able to interpolate the location of the target node. We believe that this process resulted in errors less than 1 m, 0.2 m/s, and 5° in the location, speed, and heading estimates, respectively.

Measurements: We have achieved 1.7 sec update rate in our experiments, coming from 0.3 sec coordination phase, 0.4 sec measurement time, and 1 sec time to route the measurements from the infrastructure nodes to a PC. Occasionally, measurements from some of the nodes failed to reach the PC due to the radio collisions. We have considered the measurement round invalid, if fewer than 50% of the measurements were received. The CNLS-EKF algorithm was not invoked for the invalid measurements.

We ran experiments that evaluate our algorithm in two cases: (1) an expected case where the tracked object moves along straight lines for substantial amounts of time with a constant speed, and (2) an extreme case where the tracked object changes its speed and direction abruptly and significantly after relatively short periods of time. The Kalman

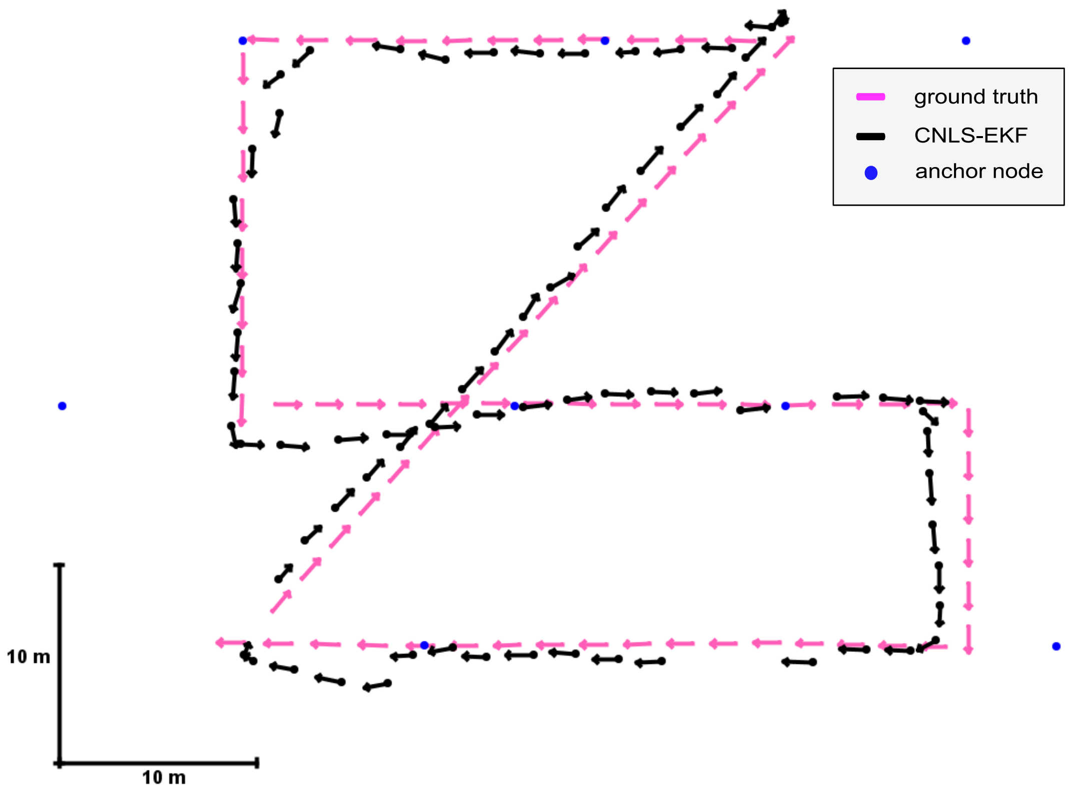

Figure III.20: Experimental evaluation of the CNLS-EKF algorithm. Eight anchor nodes are shown as black dots. The location and the velocity of the tracked node are shown as a black dot and an arrow, respectively. The ground truth is shown as gray arrows.

filter assumption of the constant speed of tracked object is violated in case (2), resulting in degraded performance.

Experiment 1: The deployment setup is shown in Fig III.20. The ground truth is shown in gray and the tracking algorithm results are indicated by the black arrows. The tracked node was moving at the mean speed of 1.3 m/s which was varying at most 0.1 m/s. We ran the CNLS-EKF algorithm on this data, setting the process noise covariance matrix Q with 0.5, the measurement noise covariance matrix R with 0.25, and the radius r of the barrier function $b(\mathbf{x})$ of 3 m. The average location, speed, and heading errors can be found in Table III.2. We also ran EKF alone on the same data set which shows that CNLS-EKF achieves only a modest accuracy improvement in this case.

Table III.2: The average errors were calculated based on 74 measurements collected during Experiment 1. The improvement over the EKF was modest.

Average Errors:	**Location**	**Speed**	**Heading**
CNLS-EKF algorithm	1.4 m	0.14 m/s	7°
Improvement over EKF	8.8%	2.4%	4.8%

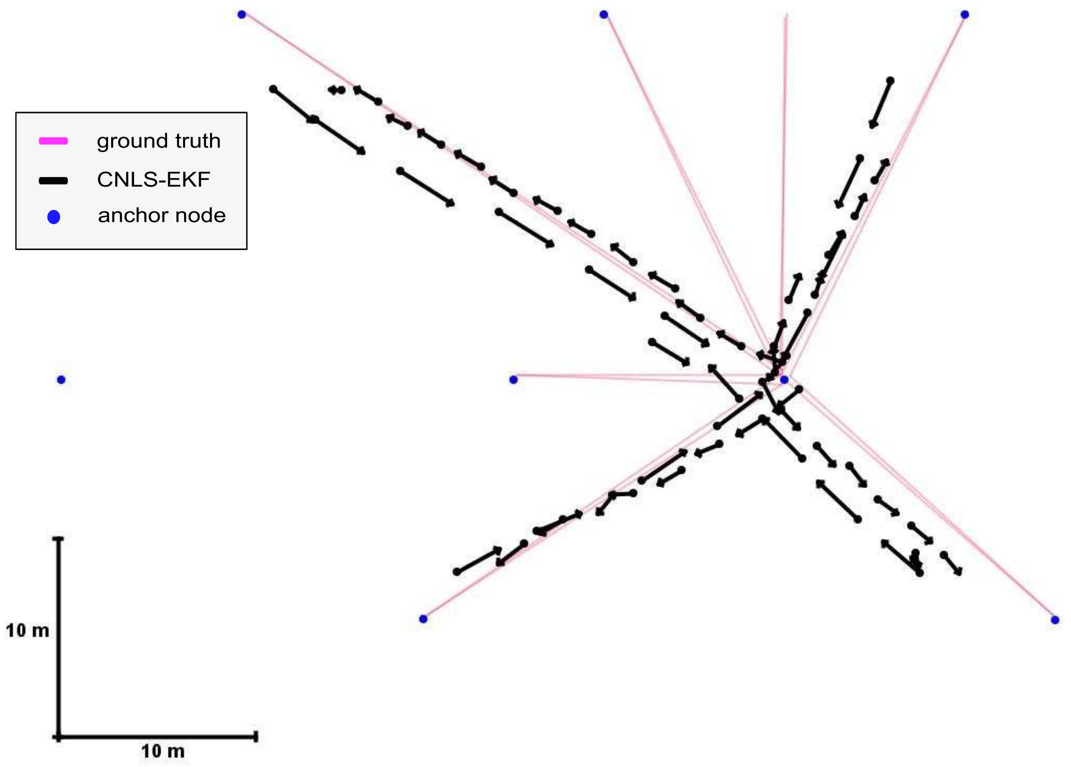

Figure III.21: We show the worst-case situation for the Kalman filter: in a star-like topology, the tracked node moved slowly away from the center and then fast towards the center of the star. Results for only four out of seven tracks are shown for clarity.

Experiment 2: We evaluated the scenario which violated the assumption of the constant velocity of the tracked node, based on which the Kalman filter was modeled. Both the speed and the heading of the tracked node was changed frequently: we selected a central point in our deployment area and designed the path to be followed by the tracked node as a star graph (see Fig III.21). A person carrying the tracked node was walking (1.2 m/s) when moving away from the center. Upon reaching the outside endpoint, the person started running (up to 3 m/s) in the opposite direction, towards the center point. We have measured 94 data points in this case and show the results in Table III.3. The improvement over EKF with no maneuver correction was more significant this time. However, the speed and the heading errors of EKF were similar to those of CNLS-EKF.

Table III.3: We collected 94 measurements in Experiment 2. The improvement over the EKF was more significant, especially in location estimation.

Average Errors:	**Location**	**Speed**	**Heading**
CNLS-EKF algorithm	2.5 m	0.36 m/s	18°
Improvement over EKF	35.8%	5.5%	0.6%

Discussion: Both EKF and CNLS-EKF algorithms perform well under the assumption of the linear dynamics of the tracked node. This is because the EKF converges to the true location of the tracked node relatively fast after the maneuver. If the maneuvers are relatively infrequent, the small additional error of the filter is averaged out for the whole track. If the tracked node changes its velocity significantly and the maneuvers are frequent, the EKF takes a long time to converge, or diverges completely. Consequently, the location error grows significantly. The constrained optimization is able to rapidly correct the Kalman filter state after a maneuver which results in faster convergence of the CNLS-EKF filter and in a better overall location accuracy. Notice that the velocity estimates have approximately the same errors for both EKF and CNLS-EKF. Intuitively, this is because our measurement model is based on estimating the relative velocities of the tracked node which gives us more information on its velocity than its location. Therefore, even if we optimize our objective function at a wrong location, the velocity estimates are still relatively accurate (see Fig. III.18).

3.9.12 Evaluation in simulation

The experimental evaluation of our tracking algorithm is a complex and time consuming process. Therefore, we implemented a simulation engine that can benchmark our tracking algorithm using thousands of measurements. The parameters of the simulation engine are:

(1) 2D coordinates of infrastructure nodes S_i,

(2) the track of the mobile node,

(3) the wavelength λ_t of the transmitted signal,

(4) σ_m standard deviation of the measurement noise,

(5) σ_f standard deviation of the change of the interference frequency $(\widehat{f})$, and

(6) the measurement update time t_m.

The engine starts the simulation using the first timestamp recorded in the track and simulates a measurement round every t_m seconds, until the last timestamp. For every measurement round, the location and the velocity of the tracked node is recalculated based on the track data, the relative speeds v_i of the tracked node with respect to S_i nodes are calculated and converted to the frequencies f_i using (III.20). Finally, f_i is corrupted with a normally distributed zero-mean noise with standard deviation σ_m. For the consecutive measurement round, the interference frequency $\widehat{f}$ used in (III.20), is corrupted by a normally distributed zero-mean noise with standard deviation σ_f to simulate its variance over time.

We have simulated both experiments described in the previous section to benchmark the simulation engine. The parameters of the simulation, $\lambda = 0.67$ cm, $\sigma_m = 0.21$ Hz, $\sigma_f = 1.0$ Hz, and $t_m = 1.7$ sec, were measured experimentally (see Section III.9.10). The average location error of the CNLS-EKF algorithm in simulation was approximately 7% better than in the experiments. A little bit better location accuracy in the simulation is expected because we did not model the routing message-loss in simulation. The speed and the heading accuracy was much better in simulation than in the experiments. We believe that this significant error increase is due to the non-zero time that the maneuver takes in the experiments as compared to the zero time in simulation. We estimate that it takes about a second for a person to stop turn around and accelerate to the desired speed. If we calculate the experimental errors without considering the measurements taken during the maneuvers, the experimental and the simulated errors align within the ground truth measurement error.

Simulated data allows us to evaluate our algorithm comprehensively using thousands of measurements under different randomly generated deployment scenarios and tracks:

Deployments. We assumed the deployment area to be a 50×30 m rectangle and randomly generated coordinates of the infrastructure nodes from a uniform distribution inside this area. To prevent degenerated cases, we placed four infrastructure nodes in the corners of the area in all deployments discussed in this section.

Tracks. We initialized the tracked node in the center of the deployment area and generated a random 200-point track T. The track is defined as a series of timestamped *(location,velocity)* pairs. The maximum speed of the tracked node v_m and the measurement time t_m are parameters of the simulation engine.

In the following three simulations, we have evaluated the tracking accuracy of the CNLS-EKF algorithm while varying the number of infrastructure nodes (receivers), the maximum speed of the tracked node and the tracking update rate.

Simulation 1: the tracking accuracy dependance on the number of infrastructure nodes was studied. We generated five tracks T_i using $v_m = 2$ m/s and $t_m = 1.7$ sec and five deployments D_i containing 12 infrastructure nodes. Next, we simulated the performance of dTrack for each deployment D_i along all five tracks (collecting 5000 measurements in total). We repeated this experiment with a decreasing number of anchors, selecting subsets of 4,6,8, and 10 nodes out of the 12 nodes in the original deployments. We show the average location, speed, and heading accuracy in Table III.4.

Simulation 2: the CNLS-EKF algorithm was evaluated for different maximum speeds of the tracked node. We used five different deployments which contained eight infrastructure nodes. Also, $t_m = 1.7$ sec stayed constant for the duration of the simulation. We varied the maximum speed of the tracked node v_m, using values $1, 2, 3$ and 5 m/s. For a given

Table III.4: Tracking accuracy vs. the number of participating infrastructure nodes. For a given number of receivers, the tracking errors were computed using 5000 randomly generated data points.

#receivers	**Location (m)**	**Speed (m/s)**	**Heading(°)**
12	1.4	0.07	5.5
10	1.6	0.08	5.8
8	1.8	0.08	6.2
6	2.3	0.09	8
4	3.3	0.09	8.6

Table III.5: Tracking accuracy vs. the maximum speed of the tracked node. For a given maximum speed v_m, the track that the node follows was generated using random speeds from the $(0,v_m)$ interval.

Speed	**Location (m)**	**Speed (m/s)**	**Heading(°)**
1 m/s	1.6	0.05	5.8
2 m/s	1.8	0.08	6.2
3 m/s	3.6	0.28	7.9
5 m/s	9.0	0.41	13.0

maximum speed, we generated five random tracks T_i, and simulated each track with each deployment (again, 5000 measurements are collected). The performance of the CNLS-EKF algorithm is shown in Table III.5.

Simulation 3: finally, we evaluated the tracking accuracy for different tracking update rates. Similarly to Simulation 2, we kept the deployments constant. Next, we kept the maximum speed constant at $v_m = 5$ m/s and simulated the tracks with update rates of once every $0.4, 0.8, 1.2$, and 1.7 seconds. Results are shown in Table III.6.

Discussion: In general, adding more receivers, limiting the maximum speed of the tracked node and increasing the temporal resolution of the collected data help to improve the accuracy of dTrack. However, improving the accuracy by increasing the number of infrastructure nodes becomes inefficient relatively soon. In fact, 8 nodes in the 1500 m^2 area seems to be a good tradeoff between the tracking accuracy and the required density of the infrastructure nodes. The algorithm quickly becomes inaccurate, if the speed of the tracked node increases, up to a point when the Kalman filter completely diverges from the true location of the tracked node. The tracking accuracy in this case can be improved by increasing the update rate of dTrack. Improving the update rate to once per 1.2 sec was sufficient to prevent the Kalman filter from diverging when the tracked node moved at 5 m/s. However, we estimate that the limitations of the current hardware would prevent us from running dTrack faster than once

Table III.6: Tracking accuracy vs. the tracking update rate. For faster update rates, the random tracks were generated with a better temporal resolution which allows for better tracking accuracy.

Update rate	**Location (m)**	**Speed (m/s)**	**Heading($^\circ$)**
0.4 sec	1.4	0.15	3.8
0.8 sec	1.8	0.17	7.1
1.2 sec	2.1	0.20	4.8
1.7 sec	9.0	0.41	13.0

every 1 sec and thus, our algorithm will not scale in the case of maneuvering nodes moving at speeds over 5 m/s.

Chapter IV

Conclusions and Future Work

In this final chapter, we summarize the contributions of this thesis and discuss potential future work. Some of our contributions have already been published in peer reviewed journals and conferences [69, 53, 52, 57, 70, 89, 54, 55].

4.1 Contributions

The contributions of this thesis are as follows: we have studied time synchronization and localization of wireless sensors and identified a number of conceptual and practical issues in the existing techniques. We have addressed these issues with novel techniques, implemented a number of efficient time synchronization and localization services that provide high levels of accuracy and cost-effective deployment in real-world applications. We have also extensively evaluated these services both in deployment experiments, and in simulation.

4.1.1 Time coordination

We have observed that existing time synchronization algorithms face problems in keeping up with hardware evolution and provide little guidance on how they should be used. We have proposed a structured approach that alleviates these problems.

Structured approach to the design of time synchronization algorithms:

We have proposed that time synchronization protocols should be designed in three layers and should use standardized interfaces for the communication among them. The layered approach improves adaptability, reusability, and portability of time synchronization protocols over different hardware platforms and allows for their seamless evolution on rapidly changing hardware platforms. In particular, the hardware abstraction layer isolates the portability-related code changes to a single layer. The implementation of this layer should be done by hardware domain experts, providing fine-tuned time synchronization performance and shielding the time synchronization developers from the intricate details of the underlying hardware. The time synchronization services layer is the core layer of each time synchronization protocol, implementing the actual time synchronization functionality. The implementation of time synchronization services is required to conform to a set of application programming interfaces (APIs). We have argued that the APIs abstract common time synchronization usage patterns found in current WSN applications and help application developers to find the best fitting protocol for their requirements.

Efficient implementation of ETA timestamping primitive:

We have proposed a radio message timestamping primitive *Elapsed Time on Arrival (ETA)* as the point of abstraction between the operating system and the time synchronization services. The ETA primitive is different from existing techniques, focusing not only on achieving high accuracy of synchronization, but achieving efficient utilization of system resources as well. We have shown that the ETA design supports portability by implementing it on multiple hardware platforms.

Quantitative comparison of multiple time synchronization protocols:

We have implemented a number of time synchronization services derived from the needs of several well-known and common WSN applications. The *Event timestamping service* allows to timestamp events in the local time of one or more sensor nodes and performs inter-node time conversions that allows a sink node to determine the event time in its own local time. The *Virtual global time service* provides a single, virtual, and globally shared timebase at each node in the network. Finally, the *coordinated action service* allows multiple nodes to schedule an event at a future time instant.

All time synchronization services were extensively evaluated in a series of experiments that involved a significant number of deployed sensor nodes. We have used the same hardware platform and similar deployments of sensor nodes in all of our experiments to allow for a meaningful comparison of the performance of the different time synchronization protocols. Comparisons of this kind are important because they can guide application developers in making informed decisions when choosing a time synchronization method for their applications.

4.1.2 Space coordination

We have observed that existing WSN localization methods have either low cost, adequate accuracy, or acceptable range, but do not satisfy all three criteria at the same time. We proposed a novel radio interferometric technique that attains high accuracy and long range simultaneously, while utilizing low-cost, low-power, off-the-shelf WSN hardware.

Radio interferometric ranging:

In comparison to traditional radio signal strength based techniques, the interferometric technique allows for larger range and higher accuracy, because it utilizes the phase of the incoming radio signal rather than its amplitude. The limited computational power of common WSN hardware and the high transmit frequency of the radio forced us to find alternatives to the direct analysis of the radio signal: we have measured the phase of the radio signal indirectly through radio interference of two simultaneously transmitting nodes. However, extremely precise synchronization between the two transmitters and the receiver would be required to correlate the transmitted and received signal phase. Therefore, we have used two receivers to measure the relative phase of the interference signal which depends only on the distances between the four nodes and the carrier wavelength.

Radio interferometric ranging allowed us to implement highly accurate localization and tracking techniques that is applicable in practical deployments because of the low required node density and no extra hardware requirements. We have evaluated the radio interferometric ranging in a rural area using 16 ExScal motes deployed on the ground in an approximately 12000 m^2 area with an average closest-neighbor distance of 35 m and a maximum node distance of 170 m. Using three of the nodes as anchors, the remaining 13 nodes were localized with 4 cm average and 12 cm maximum localization error. We have also deployed our system in the Vanderbilt football field, covering 800 m^2 area with 4 anchor nodes and tracked three mobile nodes simultaneously achieving 4.5 sec update rate and 1 m average localization error.

RF Doppler shift tracking:

Even though the radio interferometric techniques were shown to provide superior accuracy and range, they also have a few drawbacks. The main drawback is the relatively long measurement time required to achieve high localization accuracy. Consequently, the update rate of tracking, as well as the maximum allowed speed of the tracked objects, are limited.

Hence, we have proposed to utilize a similarly constructed interference signal, but instead of measuring its phase, we measured the Doppler frequency shifts at mobile nodes to estimate their *velocity and location* simultaneously. We have shown that a simpler and faster tracking algorithm can be designed this way, because only a single carrier frequency is needed.

We have used the same experimental deployment in the Vanderbilt football stadium as for the radio interferometric tracking and achieved 1.7 sec update rate and up to 3 m/s maximum speed, while attaining similar accuracy. We have estimated that current WSN hardware will allow us to achieve 1 sec update rate, in which case objects moving up to 5 m/s can be tracked.

4.2 Future Work

Time synchronization hardware abstractions:

We have built all time synchronization services on top of a single timestamping primitive, the ETA. However, a number of alternative timestamping primitives exist, such as reference broadcast or sender-receiver synchronization (see Section II.1.4). For certain hardware platforms, or in certain WSN application deployments, it may be beneficial to build the time synchronization protocols using a different timestamping primitive. For example, the target hardware platform can utilize a radio chip that does not support low level access to the transmission data buffer in which case ETA could not be implemented in the current form. Therefore, we would like to implement other primitives in our hardware abstraction layer. We believe that using our structured approach, the implementation of existing time synchronization protocols, such as RBS [20] and TPSN [28], would become less complex, more robust and more accurate than the existing implementations.

Time synchronization services:

We would like to extend the list of time synchronization services that we have described in Section II.8. For example, in monitoring applications, one or more observer nodes are periodically sampling some phenomena. The data series from each node is to be collected at a sink node for correlated analysis. This *data series timestamping service* requires timestamps on each sample that are both accurate with regards to a series (intra-series accuracy) as well as across the series (inter-series accuracy). Some of the challenges are how to compare observations when the sampling frequencies of two different data sources are not identical, the observations are collected over a (potentially long) period of time, and samples cannot be dropped or added (i.e. time must not have any sudden jumps).

Similarly, a continuous coordinated action service is often required, for example, in tracking applications, where ranging measurements need to be scheduled periodically to achieve real-time update rate of localization. Currently, time synchronization radio messages need to be broadcasted every time the coordinated action is scheduled. A more efficient implementation may utilize the fact that the coordinated action is requested periodically. Clock skews of the nodes can be estimated and a number of actions can be scheduled without requiring to transmit the time synchronization messages for each of them.

RF Doppler tracking extension:

We would like to extend the interferometric measurement as follows: instead of two nodes transmitting at the same time, we can have one node starting sooner and the second node finishing later which would allow us to measure the signal strength (RSS) of both transmitters along with the frequency and the phase of the interference signal. This way, mTrack, dTrack, and RSS based tracking techniques could be combined into a single algorithm which would be potentially much more robust and accurate.

Each receiver measures four quantities in this approach: the signal amplitudes (RSSI) of the two transmitters, and the frequency and the phase of the interference signal. The amplitudes and phase differences give us information about the location of the receiver with respect to the two transmitters, and the frequencies tell us about the velocity of the receiver. This improves the previous work in two ways: the tracked node is not required to be moving (improves dTrack), and the measurement time is relatively short because only one radio channel is used (improves mTrack). We expect that the extended Kalman filter technique would perform well for the data fusion: the location and velocity of the tracked node, and radio calibration parameters for both transmitters would be estimated by the EKF and updated based on the measurements just as in dTrack.

Appendix A

Intersection of Two Hyperbolas

The basic building block of our tracking algorithm is the analytic solver that calculates intersection of two hyperbolas. Here, we show that it is possible to find a closed form formula for the intersection if the two hyperbolas share a common focus.

We assume that the common focus of the hyperbolas is located at $(0,0)$ and another focus is aligned on the x-axis. Let hyperbola h_{AB} be defined by its foci $A = (0,0), B = (b_x, 0)$ and the distance R_{AB}, and h_{AC} be defined by $A = (0,0), C = (c_x, x_y)$, and R_{AC}. Given the coordinates of A, B, C and the distances R_{AB}, R_{AC}, we want to find the intersection points of h_{AB}, h_{AC}.

The following two equations for h_{AB} and h_{AC} can be derived using the equation defining a hyperbola in 2D plane:

$$\begin{aligned}\sqrt{x^2+y^2} - \sqrt{(x-b)^2+y^2} &= R_{AB} \\ \sqrt{x^2+y^2} - \sqrt{(x-c_x)^2+(y-c_y)^2} &= R_{AC}\end{aligned}$$

Next, subtract $\sqrt{x^2+y^2}$ and square both equations :

$$\begin{aligned}x^2+y^2-2\sqrt{x^2+y^2}R_{AB}+R_{AB}^2 &= (x-b)^2+y^2 \\ x^2+y^2-2\sqrt{x^2+y^2}R_{AC}+R_{AC}^2 &= (x-c_x)^2+(y-c_y)^2\end{aligned}$$

These equations can be simplified as

$$\begin{aligned}\sqrt{x^2+y^2}R_{AB} &= bx - b^2 + R_{AB}^2 \\ \sqrt{x^2+y^2}R_{AC} &= c_x x + c_y y - c^2 + R_{AC}^2\end{aligned}$$

and equated

$$(bx - b^2 + R_{AB}^2)R_{AC} = (c_x x + c_y y - c^2 + R_{AC}^2)R_{AB}$$

Next, we can express x from this equation and substitute it into one of the original equations, essentially getting a quadratic equation in the parameter y which has a closed form solution. Consequently, we can find a closed formulae for both x and y, which are the intersection coordinates of the two hyperbolas h_{AB} and h_{AC}. See [56] for the derivation of the complete formulae.

Appendix B

Mobility Error Approximation

The location of the tracked node can change significantly during the ranging phase and robust tracking algorithms should compensate for this velocity related location change. Interferometric tracking only approximates the velocity related errors (Section III.8.4). Here, we show that the error of this approximation is small.

The error of the approximation was defined as $\varepsilon(\alpha) = d_{AX'} - d_{AX} - x_a$ (see Figure III.11). From the definition of cosine, and the law of cosines, we know that

$$\begin{aligned} x_a &= d_{XX'} \cos\alpha, \\ d_{AX'} &= \sqrt{d_{AX}^2 + d_{XX'}^2 + 2d_{AX}d_{XX'}\cos\alpha}\ . \end{aligned}$$

The extremes of $\varepsilon(\alpha)$ are found by differentiating x_a and $d_{AX'}$ by α:

$$\begin{aligned} \frac{\partial x_a}{\partial\alpha} &= -d_{XX'}\sin\alpha, \\ \frac{\partial d_{AX'}}{\partial\alpha} &= \frac{-d_{AX}d_{XX'}\sin\alpha}{\sqrt{d_{AX}^2 + d_{XX'}^2 + 2d_{AX}d_{XX'}\cos\alpha}} \end{aligned}$$

Next we differentiate $\varepsilon(\alpha)$:

$$\frac{\partial\varepsilon(\alpha)}{\partial\alpha} = \frac{-d_{AX}d_{XX'}\sin\alpha}{\sqrt{d_{AX}^2 + d_{XX'}^2 + 2d_{AX}d_{XX'}\cos\alpha}} + d_{XX'}\sin\alpha$$

Finally, we let $\frac{\partial(\varepsilon(\alpha))}{\partial\alpha} = 0$. It is easy to see that either $\sin\alpha = 0$, or:

$$d_{AX} = \sqrt{d_{AX}^2 + d_{XX'}^2 + 2d_{AX}d_{XX'}\cos\alpha}$$

from where $\cos\alpha = -d_{XX'}/(2d_{AX})$.

To summarize, the function $\varepsilon(\alpha)$ has extremes at the following points 0, π, and $\arccos(-d_{XX'}/(2d_{AX}))$. Since $\varepsilon(\alpha) = 0$ for the first two values, we conclude that $\varepsilon(\alpha)$ has its maximum at $\arccos(-d_{XX'}/(2d_{AX}))$. Interestingly, the approximation error happens to be maximal when $d_{AX} = d_{AX'}$.

Appendix C

Jacobians

Jacobian for dTrack:

In the dTrack algorithm, we track a moving node T by observing the Doppler shifts of its signal measured by n infrastructure nodes S_i. The parameter vector $\mathbf{x} = (x, y, v_x, v_y, \widehat{f})^{\mathsf{T}}$, observation vector $\mathbf{c} = (f_1, f_2, \ldots, f_n)^{\mathsf{T}}$ and the objective function $F(\mathbf{x}) - \mathbf{c}$ define our optimization problem. The function F is a vector function defined by its components F_i :

$$F_i(\mathbf{x}) = \widehat{f} - \frac{1}{\lambda_t} \frac{v_x(x_i - x) + v_y(y_i - y)}{\sqrt{(x_i - x)^2 + (y_i - y)^2}},$$

where (x_i, y_i) are the coordinates of the node S_i.

The Jacobian $J(\mathbf{x})$ of $F(\mathbf{x}) - \mathbf{c}$ is required by both the CNLS and EKF techniques. Since $\mathbf{c}$ is constant, we calculate the Jacobian as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_1}{\partial y} & \frac{\partial F_1}{\partial v_x} & \frac{\partial F_1}{\partial v_y} & \frac{\partial F_1}{\partial \widehat{f}} \\ \vdots & & \ddots & & \vdots \\ \frac{\partial F_n}{\partial x} & \frac{\partial F_n}{\partial y} & \frac{\partial F_n}{\partial v_x} & \frac{\partial F_n}{\partial v_y} & \frac{\partial F_n}{\partial \widehat{f}} \end{bmatrix}$$

We only have to calculate five partial derivatives for the five different columns, as different lines in the Jacobian matrix $\mathbf{J}$ are computed analogously:

$$\begin{aligned}
\frac{\partial F_i}{\partial x} &= -\tfrac{1}{\lambda_t} \frac{(y_i - y)(v_y(x_i - x) - v_x(y_i - y)}{((x_i - x)^2 + (y_i - y)^2)^{3/2}}, \\
\frac{\partial F_i}{\partial y} &= -\tfrac{1}{\lambda_t} \frac{(x_i - x)(v_x(y_i - y) - v_y(x_i - x)}{((x_i - x)^2 + (y_i - y)^2)^{3/2}}, \\
\frac{\partial F_i}{\partial v_x} &= -\tfrac{1}{\lambda_t} \frac{(x_i - x)}{\sqrt{(x_i - x)^2 + (y_i - y)^2}}, \\
\frac{\partial F_i}{\partial v_y} &= -\tfrac{1}{\lambda_t} \frac{(y_i - y)}{\sqrt{(x_i - x)^2 + (y_i - y)^2}}, \\
\frac{\partial F_i}{\partial \widehat{f}} &= 1.
\end{aligned}$$

Note that the location (x, y) and the velocity vector (v_x, v_y) of the tracked node are given by the parameter vector $\mathbf{x}$; the coordinates (x_i, y_i) of the infrastructure nodes S_i are the known quantities; and observations f_i come from the observation vector $\mathbf{c}$.

Jacobian for CNLS:

The following barrier function $b(\mathbf{x})$ can be defined to constrain the location of the optimization region to a disk centered at $X_0 = (x_0, y_0)$, with radius r

$$\mathbf{b}(\mathbf{x}) = -\log(r - \|XX_0\|),$$

where $\|XX_0\| = \sqrt{(x-x_0)^2 + (y-y_0)^2}$ is the distance of $X = (x, y)$ from X_0. The Jacobian matrix $B \in \mathcal{R}^{1\times 5}$ of $b(\mathbf{x})$ is

$$B(\mathbf{x}) = \begin{bmatrix} \frac{\partial b}{\partial x} & \frac{\partial b}{\partial y} & \frac{\partial b}{\partial v_x} & \frac{\partial b}{\partial v_y} & \frac{\partial b}{\partial \widehat{f}} \end{bmatrix},$$

where

$$\begin{aligned}
\frac{\partial b}{\partial x} &= \frac{1}{r - \|XX_0\|} . \frac{x - x_0}{\|XX_0\|}, \\
\frac{\partial b}{\partial y} &= \frac{1}{r - \|XX_0\|} . \frac{y - y_0}{\|XX_0\|}, \\
\frac{\partial b}{\partial v_x} &= \frac{\partial b}{\partial v_y} = \frac{\partial F_i}{\partial \widehat{f}} = 0.
\end{aligned}$$

References

[1] I. F. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: research challenges. *Journal of Ad Hoc Networks*, pages 257–279, 2005.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, Mar. 2002.

[3] J. Aslam, Z. Butler, V. Crespi, G. Cybenko, and D. Ru. Tracking a moving object with a binary sensor network. *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2003.

[4] P. Bahl and V. N. Padmanabhan. Radar: An in-building RF-based user-location and tracking system. *IEEE Conference on Computer Communications (INFOCOM)*, 2, Mar. 2000.

[5] P. Basu and T. D. C. Little. Networked parking spaces: architecture and applications. *Vehicular Technology Conference*, 2002.

[6] M. Bauer, L. Jendoubi, and O. Siemoneit. Smart factory - mobile computing in production environments. *Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2004.

[7] B. A. Block. Physiological ecology in the 21st century: Advancements in biologging science. *Integrative and Comparative Biology Journal*, 45(2):305–320, 2005.

[8] R. R. Brooks, C. Griffin, and D. S. Friedlander. Self-organized distributed sensor network entity tracking. *Int'l Journal of High Performance Computing Applications*, 16(3), 2002.

[9] R. R. Brooks, P. Ramanthan, and A. Sayeeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, pages 1163–1171, Aug. 2003.

[10] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low-cost outdoor localization for very small devices. *IEEE Personal Communications*, 7(5):28–34, Oct. 2000.

[11] Z. Butler, P. Corke, R. Peterson, and D.Rus. Networked cows: Virtual fences for controlling cows. *Workshop on Applications of Mobile Embedded Systems (WAMES)*, 2004.

[12] Chipcon AS, CC1000: Single chip very low power RF transceiver. `http://www.chipcon.com`, 2004.

[13] Chipcon AS, CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF transceiver. `http://www.chipcon.com`, 2004.

[14] K. Chintalapudi, R. Govindan, G. Sukhatme, and A. Dhariwal. Ad-hoc localization using ranging and sectoring. *IEEE Conference on Computer Communications (INFOCOM)*, 2004.

[15] C.-Y. Chong and S. P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[16] S. Coleri, S. Cheung, and P. Varaiya. Sensor networks for monitoring traffic. *In Annual Allerton Conference on Communication, Control, and Computing*, Sept. 2004.

[17] K. A. Delin, R. P. Harvey, N. A. Chabot, S. P. Jackson, M. Adams, D. W. Johnson, and J. T. Britton. Sensor web in antarctica: Developing an intelligent, autonomous platform for locating biological flourishes in cryogenic environments. *Lunar and Planetary Science Conference*, Mar. 2003.

[18] E. W. Dijkstra. The structure of the THE multiprogramming system. *Communications of the ACM*, 11(5):341–346, May 1968.

[19] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. *Symposium on Information Processing in Sensor Networks (IPSN) SPOTS track*, Apr. 2005.

[20] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, 2002.

[21] J. Elson and K. Römer. Wireless sensor networks: a new regime for time synchronization. *ACM SIGCOMM Computer Communication Review*, 33(1):149–154, Jan. 2003.

[22] M. Endler. Large scale body sensing for infectious disease control. *in EWSN, Sentient Future Competition*, 2006.

[23] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *Pervasive Computing*, Jan. 2002.

[24] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. *ACM Conference on Mobile Computing and Networking (MOBICOM)*, pages 263–270, 1999.

[25] L. Evers, W. Bach, D. Dam, M. Jonker, H. Scholten, and P. Havinga. An iterative quality-based localization algorithm for ad hoc networks. *Int. Conf. on Pervasive Computing*, pages 55–61, Aug. 2002.

[26] C. L. Fok, G. C. Roman, and C. Lu. Rapid development and flexible deployment of adaptive wireless sensor network applications. *Int'l Conference on Distributed Computing Systems (ICDCS)*, 2005.

[27] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[28] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2003.

[29] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: a holistic approach to networked embedded systems. *Programming Language Design and Implementation (PLDI)*, June 2003.

[30] I. Getting. The global positioning system. *IEEE Spectrum 30*, page 3647, Dec. 1993.

[31] L. Girod, M. Lukac, V. Trifa, and D. Estrin. The design and implementation of a self-calibrating acoustic sensing platform. *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2006.

[32] S. D. Glaser. Some real-world applications of wireless sensor nodes. *SPIE Symposium on Smart Structures and Materials*, Mar. 2004.

[33] J. Greunen and J. Rabaey. Lightweight time synchronization for sensor networks. *Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 11–19, Sept. 2003.

[34] G.Wener-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor networks. *European Workshop on Wireless Sensor Networks (EWSN)*, 2005.

[35] A. Harter, A. Hopper, P. Stegglesand, A. Ward, and P. Webster. The anatomy of a context-aware application. *In Mobile Computing and Networking*, page 5968, 1999.

[36] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization and its impact on large scale sensor networks. *ACM Transactions on Embedded Computing System (TECS)*, 2006.

[37] T. He, S. Krishnamurthy, J. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. An energy-efficient surveillance system using wireless sensor networks. *Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2004.

[38] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[39] J. Hightower, R. Want, and G. Borriello. SpotON: An indoor 3D location sensing technology based on RF signal strength. *UW CSE 00-02-02*, Feb. 2000.

[40] J. Hill and D. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.

[41] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, pages 93–104, 2000.

[42] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, Nov. 2000.

[43] L. E. Holmquist, H. W. Gellersen, G. Kortuem, S. Antifakos, F. Michahelles, B. Schiele, and M. B. R. Maze. Building intelligent environments with Smart-Its. *Computer Graphics and Applications*, 24(1), 2004.

[44] P. Huang and B. Krishnamachari. Analysis of existing approaches and a new hybrid strategy for synchronization in sensor networks. *EmNets*, May 2006.

[45] L. G. J. Shin and F. Zhao. Distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks. *Symposium on Information Processing in Sensor Networks (IPSN)*, pages 223–238, 2003.

[46] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. *Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, pages 96–107, 2002.

[47] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering*, 1960.

[48] C. Kappler and G. Riegel. A real-world, simple wireless sensor network for monitoring electrical energy consumption. *Lecture Notes in Computer Science: Wireless Sensor Networks*, 2920/2004, 2004.

[49] S. Kim. Structural health monitoring of the golden gate bridge. `www.cs.berkeley.edu/~binetude/ggb`, 2006.

[50] H. Kopetz and W. Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, C-36(8):933–939, Aug. 1987.

[51] M. Kushwaha, K. Molnár, J. Sallai, P. Völgyesi, M. Maróti, and A. Lédeczi. Sensor node localization using mobile acoustic beacons. *MASS*, Nov. 2005.

[52] B. Kusý, G. Balogh, A. Lédeczi, J. Sallai, and M. Maróti. inTrack: High precision tracking of mobile sensor nodes. *European Conference on Wireless Sensor Networks (EWSN)*, Jan. 2007.

[53] B. Kusý, G. Balogh, P. Völgyesi, J. Sallai, A. Nádas, A. Lédeczi, M. Maróti, and L. Meertens. Node-density independent localization. *Information Processing in Sensor Networks (IPSN) SPOTS Track*, Apr. 2006.

[54] B. Kusý, P. Dutta, P. Levis, M. Maróti, A. Lédeczi, and D. Culler. Elapsed time on arrival: a simple and versatile primitive for canonical time synchronization services. *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, 2(1), 2006.

[55] B. Kusy, A. Ledeczi, and X. Koutsoukos. Tracking mobile nodes using RF Doppler shifts. *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2007.

[56] B. Kusy and J. Sallai. Analytical solution for radio-interferometric localization of mobile sensors. *Technical Report (ISIS-06-710), Vanderbilt University, http://www.isis.vanderbilt.edu*, Dec. 2006.

[57] B. Kusy, J. Sallai, G. Balogh, A. Ledeczi, V. Protopopescu, J. Tolliver, F. DeNap, and M. Parang. Radio interferometric tracking of mobile wireless nodes. *Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2007.

[58] Y. Kwon, K. Mechitov, S. Sundresh, and G. Agha. Resilient localization for sensor networks in outdoor environments. In *Int'l Conference on Distributed Computing Systems (ICDCS)*, 2005.

[59] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Comm. ACM*, 21(7):558–565, 1978.

[60] K. Langendoen and N. Reijer. Distributed localization in wireless sensor networks: A quantitative comparison. *Computer Networks, special issue on Wireless Sensor Networs*, Aug. 2003.

[61] S. Lanzisera, D. T. Lin, and K. Pister. Rf time of flight ranging for wireless sensor network localization. *Workshop on Intelligent Solutions in Embedded Systems (WISES)*, June 2006.

[62] A. Lédeczi, A. Nádas, P. Völgyesi, G. Balogh, B. Kusý, J. Sallai, G. Pap, S. Dóra, K. Molnár, M. Maróti, and G. Simon. Countersniper system for urban warfare. *ACM Transactions on Sensor Networks*, 1(1):153–177, Nov. 2005.

[63] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for wireless sensor networks. *Ambient Intelligence, Sprinter-Verlag*, 2005.

[64] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing*, 2002.

[65] J. P. Lynch, A. Partridge, K. H. Law, T. W. Kenny, A. S. Kiremidjian, and E. Carryer. Design of piezoresistive mems-based accelerometer for integration with wireless sensing unit for structural monitoring. *Journal of Aerospace Engineering*, 16(3):108–114, July 2003.

[66] K. Madsen, H. Nielsen, and O. Tingleff. Methods for nonlinear least squares problems. *Technical report, Technical University of Denmark*, 2004.

[67] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler. Wireless sensor networks for habitat monitoring. *Workshop on Wireless Sensor Networks and Applications (WSNA)*, Aug. 2002.

[68] M. Maróti. Directed flood-routing framework for wireless sensor networks. In *ACM/IFIP/USENIX Conference on Middleware*, pages 99–114, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[69] M. Maróti, B. Kusý, G. Balogh, P. Völgyesi, A. Nádas, K. Molnár, S. Dóra, and A. Lédeczi. Radio interferometric geolocation. *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2005.

[70] M. Maróti, B. Kusý, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 39–49, Nov. 2004.

[71] K. Martinez, J. K. Hart, and R. Ong. Environmental sensor networks. *Computer*, 37(8), 2004.

[72] D. J. McCafferty, I. L. Boyd, T. R. Walker, and R. I. Taylor. Can marine mammals be used to monitor oceanographic conditions? *Marine Biology: Biomedical and Life Sciences and Earth and Environmental Science*, 134(2), 1999.

[73] Crossbow Mica2 mote. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`.

[74] Crossbow MICAz Mote. `http://www.xbow.com/Products/productsdetails.aspx?sid=101`.

[75] F. Michahelles, T. Ahonen, and B. Schiele. A-life - increasing survival chances in avalanches by wearable sensors. *Workshop on Smart Appliances and Wearable Computing (IWSAWC)*, 2003.

[76] D. L. Mills. Internet time synchronization: The network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, Oct. 1991.

[77] D. L. Mills. Improved algorithms for synchronizing computer network clocks. *IEEE/ACM Transactions on Networks*, 3(3):245–254, June 1995.

[78] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2004.

[79] D. Niculescu and B. Nath. Ad hoc positioning system (aps) using aoa. *IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2003.

[80] D. Niculescu and B. Nath. Dv based positioning in ad hoc networks. *Kluwer Journal of Telecommunication Systems*, pages 267–280, 2003.

[81] S. Pattem, S. Poduri, and B. Krishnamachari. Energy-quality tradeoffs for target tracking in wireless sensor networks. *Symposium on Information Processing in Sensor Networks (IPSN)*, 2003.

[82] N. Patwari, A. O. H. III, a. N. C. M. Perkins, and R. J. ODea. Relative location estimation in wireless sensor networks. *IEEE Trans. on Sig. Proc. Special Issue on Signal Processing in Networking*, 51(8), Aug. 2003.

[83] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. *Symposium on Information Processing in Sensor Networks (IPSN) SPOTS track*, Apr. 2005.

[84] J. Polastre, R. Szewczyk, C. Sharp, and D. Culler. The mote revolution: Low power wireless sensor network devices. *A Symposium on High Performance Chips (Hot Chips 16)*, Aug. 2004.

[85] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. *ACM Conference on Mobile Computing and Networking (MOBICOM)*, Aug. 2000.

[86] J. A. Rice. Undersea networked acoustic communication and navigation for autonomous mine-countermeasure systems. *Symposium on Technology and the Mine Problem*, 2002.

[87] D. Rife and R. Boorstyn. Single tone parameter estimation from discrete-time observations. *IEEE Transactions on Information Theory*, 20(5):591–598, 1974.

[88] K. Römer. Time synchronization in ad hoc networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, ACM, Oct. 2001.

[89] J. Sallai, B. Kusý, A. Lédeczi, and P. Dutta. On the scalability of routing-integrated time synchronization. *European Workshop on Wireless Sensor Networks (EWSN)*, Feb. 2006.

[90] C. Savarese, J. Rabaey, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. *USENIX Annual Tech. Conf.*, pages 317–327, June 2002.

[91] A. Savvides, C. C. Han, and M. B. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. *ACM Conference on Mobile Computing and Networking (MOBICOM)*, pages 166–179, 2001.

[92] A. Savvides, H. Park, and M. B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. *Workshop on Wireless Sensor Networks and Applications (WSNA)*, Sept. 2002.

[93] G. Simon, M. Maróti, A. Lédeczi, G. Balogh, B. Kusý, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1–12, New York, NY, USA, 2004. ACM Press.

[94] R. Stoleru, P. Vicaire, T. He, and J. Stankovic. Stardust: A flexible architecture for passive localization in wireless sensor networks. *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2006.

[95] S. Thrun, D. Fox, W. Burgardy, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, pages 99–141, 2000.

[96] S. A. Tretter. Estimating the frequency of a noisy sinusoid by linear regression. *IEEE Transactions on Information Theory*, 31(6):832–835, 1985.

[97] P. Volgyesi, G. Balogh, A. Nadas, C. Nash, and A. Ledeczi. Shooter localization and weapon classification with soldier-wearable networked sensors. *Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2007.

[98] H. Wang, J. Elson, L. Girod, D. Estrin, and K. Yao. Target classification and localization in a habitat monitoring application. *IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, apr 2003.

[99] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, Jan. 1992.

[100] K. Whitehouse. The design of calamari: an ad-hoc localization system for sensor networks. *Master's Thesis, University of California at Berkeley*, 2002.

[101] K. Whitehouse and D. Culler. Calibration as parameter estimation in sensor networks. *Workshop on Wireless Sensor Networks and Applications (WSNA)*, Sept. 2002.

[102] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler. The effects of ranging noise on multihop localization: An empirical study. *Symposium on Information Processing in Sensor Networks (IPSN)*, Apr. 2005.

[103] K. Yedavalli, B. Krishnamachari, S. Ravula, and B. Srinivasan. Ecolocation: A technique for rf based localization in wireless sensor networks. *Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[104] D. P. Young, C. M. Keller, D. W. Bliss, and K. W. Forsythe. Ultra-wideband(uwb) transmitter location using time difference of arrival(tdoa) techniques. *Wireless Netorks*, 8(2–3), 2003.

[105] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. *UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023*, May 2001.

[106] F. Zhao, J. Shin, and J. Reic. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, Mar. 2002.